ALIBABA CLOUD

Alibaba Cloud

Apsara Stack Enterprise

User Guide - Analytics and Artificial Intelligence

Product Version: 2006, Internal: V3.12.0

Document Version: 20201020

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.MaxCompute

1.1. What is MaxCompute?

MaxCompute is a data processing platform developed by Alibaba Group to process large volumes of data. MaxCompute provides channels for upload and download, a range of computing and analysis features including SQL and MapReduce, and comprehensive security solutions.

MaxCompute is used to store and compute large volumes of structured data. It provides warehouse solutions for large amounts of data, as well as big data analysis and modeling services.

As data collection techniques are becoming increasingly diverse and comprehensive, industries are amassing larger volumes of data. The scale of data collection has increased from 100 GB to over 1 PB, far exceeding the processing capabilities of traditional software. Analysis tasks for large volumes of data require distributed computing instead of reliance on a single server. However, distributed computing models require skilled data analysts. To use a distributed model, data analysts must understand the business needs and underlying computing model.

MaxCompute is designed to provide an intuitive approach to analyze and process large amounts of data without the need for distributed computing knowledge. MaxCompute is widely implemented within Alibaba's businesses for scenarios such as data warehousing and BI analysis for large Internet enterprises, website log analysis, e-commerce transaction analysis, and exploration of user characteristics and interests.

MaxCompute provides the following features:

- Data channel
  - Tunnel: provides highly-concurrent offline data upload and download services. MaxCompute Tunnel enables you to upload or download a large volume of data to or from MaxCompute. You must use a Java programming API to access MaxCompute Tunnel.
  - DataHub: provides real-time upload and download services. Data uploaded through DataHub is available immediately, while data uploaded through MaxCompute Tunnel is not.
- Computing and analysis
  - SQL: MaxCompute stores data in tables and provides SQL query capabilities. MaxCompute can be used as traditional database software, but it is far more powerful and is capable of processing petabytes of data. MaxCompute SQL does not support transactions, indexes, or operations such as UPDATE and DELETE. The SQL syntax used in MaxCompute is different from that in Oracle and MySQL. SQL statements from other database engines cannot be migrated seamlessly to MaxCompute. MaxCompute SQL responds to queries within a few minutes or seconds, instead of milliseconds. MaxCompute SQL is easy to learn. You can get started with MaxCompute SQL based on your prior experience of database operations, without having a deep understanding of distributed computing.
  - MapReduce: First proposed by Google, MapReduce is a distributed data processing model that has gained extensive attention and been used in a wide range of business scenarios. This document briefly describes the MapReduce model. You must have a basic knowledge of distributed computing and relevant programming experience before using MapReduce. MapReduce provides a Java programming interface.

  - Graph: an iterative graph computing framework provided in MaxCompute. Graph computing jobs use graphs to build models. A graph is a collection of vertices and edges that have values. You can edit and evolve a graph through iteration to obtain the final result. Typical applications include PageRank, single source shortest path (SSSP) algorithm, and K-means clustering algorithm.
  - Unstructured data access and processing (integrated computing scenarios): MaxCompute SQL cannot directly process external data (such as unstructured data from OSS). Data must be imported to MaxCompute tables by using relevant tools before computation. The MaxCompute team introduces the unstructured data processing framework to the MaxCompute system architecture to resolve this problem.

    MaxCompute can process the following data sources by creating external tables:

    - Internal data sources: OSS, Table Store, AnalyticDB, ApsaraDB for RDS, HDFS (Alibaba Cloud), and TDDL.
    - External data sources: HDFS (open source), MongoDB, and Hbase.
  - Unstructured data access and processing (inside MaxCompute): By reading and writing volumes, MaxCompute can store unstructured data, which otherwise must be stored in an external storage system.

- Spark on MaxCompute: a big data analytics engine designed by Alibaba Cloud to provide big data processing capabilities for Alibaba, government agencies, and enterprises. For more information, see Spark on MaxCompute.
- Elasticsearch on MaxCompute: an enterprise-class full-text retrieval system developed by Alibaba Cloud to retrieve large volumes of data. It provides near-real-time (NRT) search performance for government agencies and enterprises. For more information, see Elasticsearch on MaxCompute.
- SDK: a toolkit provided for developers. For more information, see Java SDK.
- Security solution: MaxCompute provides powerful security services to guarantee user data security.

1.2. Usage notes

You can selectively read topics in this document based on your requirements. This topic provides reading suggestions in the document based on user skill level.

Beginners

If you are a beginner, we recommend that you read the following topics:

- What is MaxCompute: The topic provides a general introduction of MaxCompute and its core features. You can obtain general knowledge about MaxCompute.
- Quick start: The topic provides step-by-step examples and instructions on how to perform basic MaxCompute operations, such as installing and configuring the client, creating tables, granting permissions, importing and exporting data, running SQL tasks, running user-defined functions (UDFs), and running MapReduce.
- Basic concepts and common commands: The topic introduces the basic concepts of MaxCompute and common MaxCompute commands. This topic helps familiarize yourself with MaxCompute operations.
- Tools: The topic describes how to download, configure, and use common MaxCompute tools to perform data analysis.

Data analysts

If you are a data analyst, we recommend that you read the following topics:

MaxCompute SQL: The topic describes how to query and analyze large amounts of data stored in MaxCompute. This topic covers the following operations:

- Execute DDL statements CREATE, DROP, and ALERT to manage tables and partitions.
- Execute SELECT statements to select records in a table, and execute WHERE clauses to view the records that meet a specified filtering condition.
- Associate two tables through an EQUIJOIN operation.
- Execute GROUP BY statements to aggregate columns.
- Execute INSERT OVERVIEW or INSERT INTO statements to insert results into another table.
- Use built-in functions and UDFs to complete a variety of computations.
- Use UDTs to reference classes or objects of third-party languages in SQL statements to obtain data or call methods.
- Use UDJs to implement flexible cross-table and multi-table custom operations, and reduce the operations on the underlying details of the distributed system through MapReduce.
- Use the Select Transform feature to simplify the reference of script code.
- Collect table statistics and configure table lifecycles.
- Use regular expressions.

Developers

If you are an experienced developer with basic understanding of distributed computing and need to perform data analysis that cannot be implemented with SQL, we recommend that you read the following topics on advanced MaxCompute functional modules:

- MapReduce is a Java programming model provided by MaxCompute. You can use the API to write MapReduce programs and process MaxCompute data.
- MaxCompute Graph is a processing framework designed that iteratively computes and models graphs. A graph consists of vertices and edges, both of which contain values. MaxCompute Graph iteratively edits and evolves graphs to obtain analysis results.
- Eclipse plugin provides an IDE to help you complete development of MapReduce, UDFs, and Graph.
- Java SDK is a toolkit provided to developers.
- MaxCompute Tunnel allows you to perform batch upload and download operations on offline data to and from MaxCompute.

Project owners or administrators

If you are a project owner or administrator, we recommend that you read the following topics:

Security solution: This topic describes how to authorize users, enable cross-project resource sharing, configure project data protection, and configure authorization policies.

1.3. Preparations

1.3.1. Log on to the ASCM console

This topic describes how to log on to the Apsara Stack Cloud Management (ASCM) console.

Prerequisites

- The IP address or domain name of the ASCM console is obtained from deployment personnel. The URL used to log on to the ASCM console is in the following format: https://IP address or domain name of the ASCM console.
- A browser is available. We recommend that you use Google Chrome.

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press Enter.
2. Enter the correct username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password as prompted. For security purposes, your password must meet the complexity requirements. Specifically, the password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as ! @ # $ %

3. Click **Login** to go to the homepage of the ASCM console.

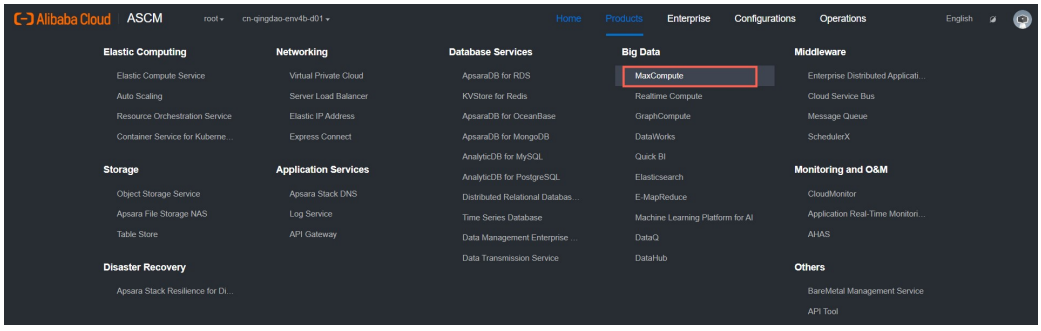
1.3.2. Create an Apsara Stack tenant account

This topic describes how to create an Apsara Stack tenant account.

Procedure

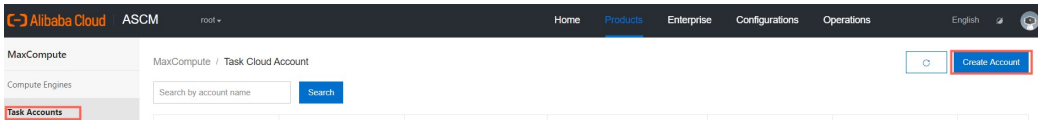
1. Log on to the Apsara Stack Cloud Management (ASCM) console as an administrator.
2. In the top navigation bar, choose **Products** > **Big Data** > **MaxCompute** to go to the **MaxCompute** homepage.

   Go to the MaxCompute homepage



3. In the left-side navigation pane, click **Task Accounts**. In the upper-right corner of the page that appears, click **Create Account**.

   Create an account



4. In the dialog box that appears, specify the required parameters and click **OK**.

Account configurations

Parameter	Description
Name	The name of the Apsara Stack tenant account.
Organization	The organization to which the Apsara Stack tenant account belongs.
Description	The description of the Apsara Stack tenant account. You can leave this parameter empty.

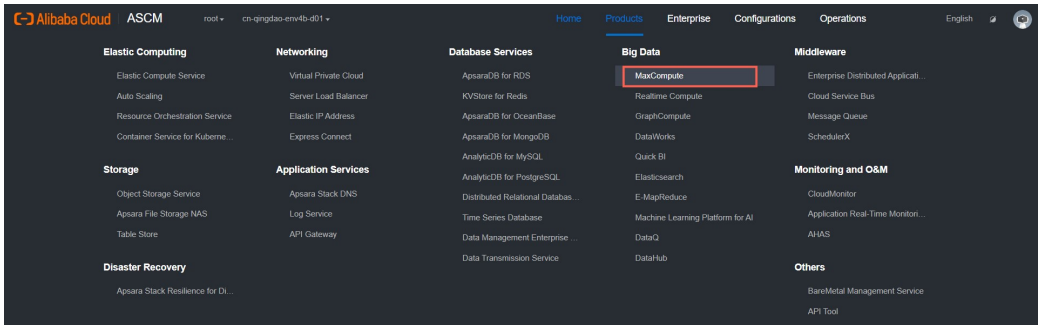
1.3.3. Create a project

This topic describes how to create a MaxCompute project.

Procedure

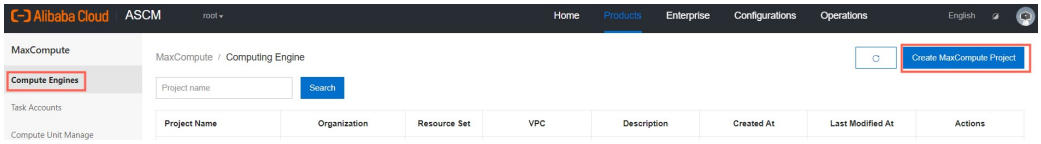
1. Log on to the Apsara Stack Cloud Management (ASCM) console as an administrator.
2. In the top navigation bar, choose **Products** > **Big Data** > **MaxCompute** to go to the **MaxCompute** homepage.

   Go to the MaxCompute homepage



3. In the left-side navigation pane, click **Computing Engines**. In the upper-right corner of the page that appears, click **Create MaxCompute Project**.

   Create a MaxCompute project



4. On the page that appears, specify the required parameters and click **Submit**.

Region configurations

Parameter	Description
Organization	The organization that you select for the project.
Resource Set	The resource set in the selected organization.
Region	The region where the cluster is deployed.
VPC	The Virtual Private Cloud (VPC) in the region. The VPC is used for network isolation.
Cluster	The information about the cluster of the project.
Encryption Settings	No is selected for Encrypt by default. If you select Yes for Encrypt, the available encryption algorithms are AES-CTR, AES-256, and RC4.
Encryption Key	If you select Yes for Encrypt, this parameter is available. The default value is Generate Automatically.

Resource configurations

Parameter	Description
Resource Group	The resource group in the selected cluster. If no resource groups are available, create one.
Apply (GB)	The storage space that is requested for the project. The value cannot be greater than the total capacity of the disk.

Basic configurations

Parameter	Description
MaxCompute Project Name	The name of the project.
Owner Account	The name of the Apsara Stack tenant account that corresponds to the project owner. The value of this parameter depends on the associated organization. You cannot specify this parameter.
Access ID	The AccessKey ID of the Apsara Stack tenant account that corresponds to the project owner. You cannot specify this parameter.

Parameter	Description
Access Key	The AccessKey secret of the Apsara Stack tenant account that corresponds to the project owner. You cannot specify this parameter.
Task Account	Optional. The account of the task.
Access ID	The AccessKey ID of the task account. You cannot specify this parameter.
Access Key	The AccessKey secret of the task account. You cannot specify this parameter.

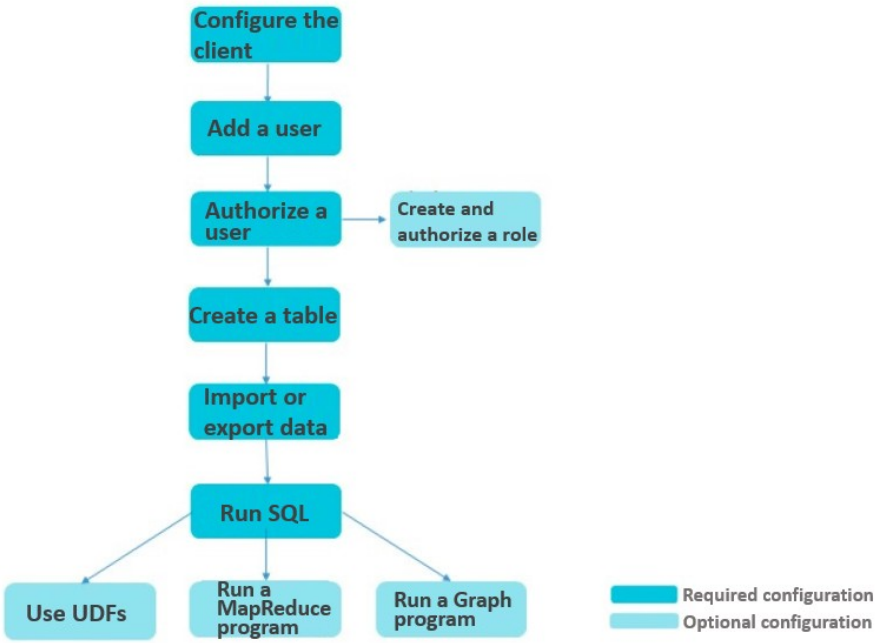
1.4. Quick start

1.4.1. Overview

This topic describes the operation process of MaxCompute. It aims to provide you with step-by-step instructions on basic MaxCompute operations.

MaxCompute operation process shows the procedure.

MaxCompute operation process



1. Configure the client.

   You must install and configure the client to use all the features of MaxCompute.

2. Add a user.

   Except for the project owner, all users must be manually added to a project and granted permissions before they can perform operations on the project.

3. Authorize a user.

   After you add a user, you must authorize the user to perform operations on the project. A user can only perform operations on the project after the user is authorized.

4. (Optional) Create and authorize a role.

   It can be very time-consuming to individually authorize users if a project contains a large number of users. Project administrators can use roles to grant users a specified set of permissions. After you authorize a role, all users assigned this role are granted the same permissions.

5. Create a table.

   After you are added to a project and authorized, you can start to use MaxCompute. Table operations are the most basic operations in MaxCompute.

6. Import or export data.

   You can use the SDK provided by MaxCompute Tunnel to compile your own Java tools and import and export data.

7. Run SQL.

   Only the limits on a few common SQL statements are described here. For more information about how to execute SQL statements, see MaxCompute SQL.

8. Then, you can perform any of the following optional operations:
   - Use UDFs.

     After you install the MaxCompute client, you can try to use UDFs. MaxCompute provides three types of UDFs: UDFs, UDAFs, and UDTFs. These functions are collectively known as UDFs.

   - Run a MapReduce program.

     After you install the MaxCompute client, you can run a MapReduce program.

   - Run a Graph program.

     After you install the MaxCompute client, you can run a Graph program.

1.4.2. Configure the MaxCompute client

The MaxCompute client allows you to access MaxCompute projects and use MaxCompute features. This topic describes how to install, configure, and run the client.

Context

The client is developed in Java. Make sure that you have JRE 1.8 installed on your local PC. In addition, make sure that you have an Apsara Stack tenant account and have obtained the AccessKey ID and AccessKey secret.

> **Note** Before you configure the client, make sure that a project is created and the AccessKey ID and AccessKey secret are obtained.

Procedure

1. Download the client package to your computer.

2. Decompress the package. The package contains the following four folders:

```
bin/
conf/
lib/
plugins/
```

3. Edit the following information in the odps_config.ini file under the conf folder:

```
project_name=my_project
access_id=********************
access_key=**********************
end_point= <Endpoint of MaxCompute>
```

> **Note**
> - Set access_id to the AccessKey ID and access_key to the AccessKey secret of your Apsara Stack tenant account.
> - project_name=my_project specifies the project that you want to access. This is the default project that is accessed each time you log on to the client. If this parameter is not specified, you must run the `use project_name` command to access the project after you log on to the client.
> - Set end_point to the endpoint of MaxCompute. The endpoint varies based on the user account.
> - For more information about the client, see MaxCompute client.

4. After the modification, run the *./bin/odpscmd* program in a Linux system or *./bin/odpscmd.bat* in a Windows system in the bin directory to execute SQL statements. Example:

```
create table tbl1(id bigint);
insert overwrite table tbl1 select count(*) from tbl1;
select 'welcome to MaxCompute!' from tbl1;
```

> **Note** For more information about SQL statements, see MaxCompute SQL.

1.4.3. Add and delete users

Other than the project owner, all other users must be added to a MaxCompute project and granted the corresponding permissions before they can perform any operations on the project. This topic describes how a project owner can add or delete users in a project.

If you are a project owner, we recommend that you read this topic in full. If you are a common user, we recommend that you submit an application to a project owner to join a project, and read the subsequent topics once you are added to the project.

Add users

Run the following command to add a user:

```
ADD USER <full_username>;
```

Run the following command on the client to add a user (bob@aliyun.com) to a project:

```
add user bob@aliyun.com;
```

If you are not sure whether the user is already in the project, run the following command to view the users in the project:

```
LIST USERS;
```

> Note
> - After a user is added to a MaxCompute project, the user must be granted permissions by the project owner. Then, the user can perform operations authorized by the permissions.
> - For more information about authorization, see Grant and view permissions.

Delete users

Run the following command to delete a user:

```
REMOVE USER <full_username>;
```

Run the following command on the client to delete a user from a project:

```
remove user bob@aliyun.com;
```

> Note
> - Before you delete a user, make sure that you have revoked all the permissions of the user. If you delete a user without revoking the permissions of the user, the permissions are retained. If the user is added to the project again, the user will have the permissions that were granted previously.
> - For more information about how to add or delete users, see Manage users in a project.

1.4.4. Grant and view permissions

1.4.4.1. Overview

After you add a user, you need to authorize the user. A user can only perform operations on the project after the user is authorized.

Authorization is a process of granting the permission to perform an operation (such as reading, writing, or viewing), on objects (such as tables, tasks, and resources) in MaxCompute.

This topic is intended for project administrators. If you are a regular MaxCompute user, verify that you have obtained the required permissions. You can quickly skim this topic.

MaxCompute provides two authorization mechanisms, ACL authorization and policy authorization.

1.4.4.2. ACL authorization

This topic describes the commands for ACL authorization and provides examples.

ACL authorization in MaxCompute applies to the following objects: project, table, function, resource, instance, and task. Every object has different operation permissions. For more information, see ACL authorization actions.

Command syntax:

```
GRANT privileges ON project_object TO project_subject
REVOKE privileges ON project_object FROM project_subject
privileges ::= action_item1, action_item2, ...
project_object ::= PROJECT project_name | TABLE schema_name |
INSTANCE inst_name | FUNCTION func_name |
RESOURCE res_name | JOB job_name
project_subject ::= USER full_username | ROLE role_name
```

> **Note** You can skip the ROLE clause in the preceding command. It is described in the topics after this.

Example:

```
grant CreateTable on PROJECT $user_project_name to USER ALIYUN$bob@aliyun.com;
-- Grant bob@aliyun.com the permissions to create tables in the project named $user_project_name.
grant Describe on Table $user_table_name to USER ALIYUN$bob@aliyun.com;
-- Grant bob@aliyun.com the permissions to obtain information (Describe permission) in the table nam
ed $$user_table_name.
grant Execute on Function $user_function_name to USER ALIYUN$bob@aliyun.com;
-- Grant bob@aliyun.com the permissions to run the function named $user_function_name.
```

1.4.4.3. Policy authorization

This topic describes the commands for policy authorization and provides an example.

Policy authorization is a principal-based process. For more information, see Authorization policies.

Command syntax:

```
GET POLICY;
PUT POLICY <policyFile>;
GET POLICY ON ROLE <roleName>;
PUT POLICY <policyFile> ON ROLE <roleName>;
```

Example:

```
{
"Version": "1",
"Statement": [
{
"Effect": "Deny",
"Action": [
"odps:*"
],
"Resource": "acs:odps:*:projects/$user_project_name/tables/*",
 "Condition": {
"StringEquals": {
"odps:TaskType": [
"DT"
]
}
}
},
{
"Effect": "allow ",
 "Action": [
"odps:List",
"odps:Read",
"odps:Describe",
"odps:Select"
],
"Resource": [
 "acs:odps:*:projects/$user_project_name/tables/a",
"acs:odps:*:projects/$user_project_name"
],
"Condition": {
 "StringEquals": {
"odps:TaskType": [
            "SQL"
             ]
       }
  }
}
]
}
```

> **Note** The preceding example disables the Tunnel feature of $user_project_name, and grants a user the permissions to perform list, read, describe, and select operations on the project and table a in the project.

1.4.4.4. View permissions

You can run a command to view user permissions in MaxCompute.

Run the following command to view the permissions of a user:

```
show grants for $user_name;
```

> **Note** For more information about how to view user permissions, see View permissions.

1.4.5. Create and authorize a role

If a project has a large number of users, the authorization process is time-consuming. Project administrators can use roles to categorize users with the same permissions. After you authorize a role, all users assigned with this role are granted the same permissions. This topic describes how to create a role and grant permissions to it.

One user can have multiple roles, and multiple users can have the same role.

Create a role

Run the following command to create a role:

```
CREATE ROLE <roleName>;
```

Example:

```
create role player;
```

Add a user to a role

Run the following command to add a user to a role:

```
GRANT <roleName> TO <full_username>;
```

Example:

```
grant player to bob@aliyun.com;
```

Delete a role

Run the following command to delete a role:

```
DROP ROLE <roleName>;
```

Example:

```
drop role player;
```

> **Note** Before you delete a role, make sure that all users have been removed from the role.

Grant permissions to a role

The command for granting permissions to a role is similar to the command for granting permissions to a user. For more information about how to grant permissions to a user, see Grant and view permissions. For more information about role authorization, see Role management.

1.4.6. Create or delete a table

1.4.6.1. Create a table

This topic describes how to create a table.

Run the following command to create a table:

```
CREATE TABLE [IF NOT EXISTS] table_name
[(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment]
[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)] [LIFECYCLE days]
[AS select_statement]
CREATE TABLE [IF NOT EXISTS] table_name
LIKE existing_table_name
```

Example:

```
create table test1 (key string);
-- Create a non-partitioned table.
create table test2 (key bigint) partitioned by (pt string, ds string);
-- Create a partitioned table.
create table test3 (key boolean) partitioned by (pt string, ds string) lifecycle 100;
-- Create a table with a lifecycle.
create table test4 like test3;
-- Table test3 has the same attributes (such as the field type and partition type) as those of test4, ex
cept for lifecycle.
create table test5 as select * from test2;
-- Create table test5 without replicating the partition and lifecycle information of test2 to it. Only data
of test2 is copied to test5.
```

You can set partitions or lifecycles for MaxCompute tables. For more information about how to create a table, see Create a table. For more information about how to modify partitions, see Add a partition and Delete a partition. For more information about how to modify the lifecycle, see Modify the lifecycle of a table.

1.4.6.2. Obtain table information

This topic describes how to obtain the table information.

Command syntax:

```
desc <table_name>;
```

Example:

```
desc test3;
-- Obtain the information about test3.
desc test4;
-- Obtain the information about test4.
```

1.4.6.3. Delete a table

This topic describes how to delete a table.

Run the following command to delete a table:

```
DROP TABLE [IF EXISTS] table_name;
```

Example:

```
drop table test2;
```

> **Note** For more information, see Delete a table.

1.4.7. Import or export data

You can compile your own Java tools by using the SDK provided by MaxCompute Tunnel to import or export data. For the sample code, see Tunnel SDK examples.

1.4.8. Run SQL

1.4.8.1. Overview

This topic describes the limits to a few common SQL statements only. For more information about how to run SQL statements, see MaxCompute SQL.

Note the following issues when using MaxCompute SQL:

- MaxCompute SQL does not support transactions, indexes, or operations such as UPDATE and DELETE.
- The SQL syntax of MaxCompute is different from that of Oracle or MySQL. You cannot seamlessly migrate SQL statements from other database engines to MaxCompute.
- MaxCompute SQL does not respond to queries in real time. It requires a few minutes to return query results, instead of seconds or milliseconds.

1.4.8.2. SELECT statement

This topic describes limits of the SELECT statement.

The following limits apply to the SELECT statement:

- The key of the GROUP BY statement can be the name of a column in the input table, or the expression composed of input table columns. However, it cannot be the output column of the SELECT statement.

```
select substr(col2, 2) from tbl group by substr(col2, 2);
- Allowed: The key of the GROUP BY statement is an expression of columns in the input table.
select col2 from tbl group by substr(col2, 2);
-- Not allowed: The key of the GROUP BY statement is not included in columns of the SELECT statem
ent.
select substr(col2, 2) as c from tbl group by c;
-- Not allowed: The key of the GROUP BY statement is the alias of a column, or output column of the
SELECT statement.
```

> **Note** For SQL parsing, the GROUP BY operation is conducted before the SELECT operation, which means the GROUP BY statement can only use the column or expression of the input table as the key.

- DISTRIBUTE BY must be added in front of SORT BY.
- The key of ORDER BY/SORT BY/DISTRIBUTE BY must be the output column of SELECT

statement, or the column alias.

```
select col2 as c from tbl order by col2 limit 100
-- Not allowed: The key of the ORDER BY statement is not the output column of the SELECT stateme
nt, or the column alias.
select col2 from tbl order by col2 limit 100;
-- Allowed: If an output column of the SELECT statement does not have an alias, the column name is
used as the alias.
```

> **Note** For SQL parsing, the ORDER BY, SORT BY, and DISTRIBUTE BY operations come after the SELECT operation. Therefore, they can only accept the output columns of the SELECT statement as the key.

For more information about the SELECT statement, see SELECT statement.

1.4.8.3. INSERT statement

This topic describes the limits of the INSERT statement.

The following limits apply to INSERT statements:

- When an INSERT statement is used to insert data into a partition, the partition column cannot be included in the select list.

```
insert overwrite table sale_detail_insert partition (sale_date='2017', region='china') select shop_nam
e, customer_id, total_price, sale_date, region from sale_detail;
-- An error is returned. The sale_date and region columns are partition columns and are not allowed
in a INSERT statement for a static partition.
```

- When an INSERT statement is used to insert a dynamic partition, the dynamic partition column must be included in the select list.

```
insert overwrite table sale_detail_dypart partition (sale_date='2017', region) select shop_name,cust
omer_id,total_price from sale_detail;
-- An error is returned. When a dynamic partition is specified for the insert, the dynamic partition col
umns must be included among the selected columns.
```

For more information about the INSERT statement, see INSERT statement.

1.4.8.4. JOIN statement

This topic describes the limits of JOIN statements.

The following limits apply to JOIN operations:

- MaxCompute SQL supports the following JOIN operations: {LEFT OUTER|RIGHT OUTER|FULL > OUTER|INNER} JOIN.
- MaxCompute SQL supports a maximum of 128 parallel JOIN operations.

For more information about JOIN operations, see JOIN statement.

1.4.8.5. Other limits

This topic describes the other application limits of MaxCompute SQL.

- MaxCompute SQL supports a maximum of 256 concurrent UNION operations.
- MaxCompute SQL supports a maximum of 256 concurrent INSERT OVERWRITE/INTO operations.

1.4.9. Compile and use UDFs

1.4.9.1. Overview

This topic provides examples on how to compile and use MaxCompute UDFs.

MaxCompute provides three types of UDFs: UDFs, UDAFs, and UDTFs. These functions are collectively known as UDFs.

> Note
>
> - UDFs only support Java APIs. To compile a UDF program, you can upload UDF code to your project by adding resources, and run the CREATE FUNCTION statement to create a UDF.
> - This topic provides examples of UDF, UDAF, and UDTF code.

1.4.9.2. UDF example

This topic uses the convert-to-lowercase function as an example to demonstrate the process of creating a UDF. Specifically, follow these steps:

Procedure

1. Write code. To archive function, write a program and compile in terms of MaxCompute UDF frame.

   ```
   package org.alidata.odps.udf.examples; import com.aliyun.odps.udf.UDF;
   public final class Lower extends UDF { public String evaluate(String s) {
   if (s == null) { return null; } return s.toLowerCase();
   }
   }
   ```

   Name the preceding JAR package *my_lower.jar*.

2. Add resources. Specify the referenced UDF code before running UDF. User code must be added to MaxCompute by adding resources. Java UDFs must be compiled into the JAR package and added in MaxCompute as a JAR resource. The UDF framework loads the JAR package automatically and runs UDF.

   Example of the command to add JAR resources:

   ```
   add jar my_lower.jar;
   ```

> **Note** If multiple resources have the same name, rename the JAR package and modify the name of relevant JAR packages in the example command below. You can also use the "f" option to override the existing JAR resources.

3. Register the UDF. When your JAR package is uploaded, MaxCompute does not have any information about this UDF. Therefore, you must register a unique function name in MaxCompute, and specify to which function and under which JAR resources this function name corresponding.

   An example of using commands to register the UDF is as follows:

   ```
   CREATE FUNCTION test_lower AS org.alidata.odps.udf.examples.Lower USING my_lower.jar;
   ```

   Example of the function used in SQL:

   ```
   select test_lower('A') from my_test_table;
   ```

1.4.9.3. UDAF example

This topic provides an example of UDAF code for your reference.

UDAFs are registered in the same way as UDFs and are used in the same way as built-in aggregate functions.

The following UDAF code is for reference only:

```
package org.alidata.odps.udf.examples;
import com.aliyun.odps.io.LongWritable; import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable; import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
/**
project: example_project
table: wc_in2
partitions: p2=1,p1=2
columns: colc,colb,cola
*/
public class UDAFExample extends Aggregator {
@Override
public void iterate(Writable arg0, Writable[] arg1) throws UDFException { LongWritable result = (LongW
ritable) arg0;
for (Writable item : arg1) { Text txt = (Text) item;
result.set(result.get() + txt.getLength());
}
}
@Override
public void merge(Writable arg0, Writable arg1) throws UDFException { LongWritable result = (LongWrit
able) arg0;
LongWritable partial = (LongWritable) arg1; result.set(result.get() + partial.get());
}
@Override
public Writable newBuffer() { return new LongWritable(0L);
}
@Override
public Writable terminate(Writable arg0) throws UDFException { return arg0;
}
}
```

1.4.9.4. UDTF example

This topic provides an example of UDTF code for your reference.

UDTFs are registered and used in the similar way to UDFs.

UDTF code example:

```
package org.alidata.odps.udtf.examples;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDTFCollector; import com.aliyun.odps.udf.annotation.Resolve; import com.
aliyun.odps.udf.UDFException;
// TODO define input and output types, e.g., "string,string->string,bigint".
@Resolve({"string,bigint->string,bigint"}) public class MyUDTF extends UDTF {
@Override
public void process(Object[] args) throws UDFException { String a = (String) args[0];
Long b = (Long) args[1];
for (String t: a.split("\\s+")) { forward(t, b);
}
}
}
```

1.4.10. Compile and run a MapReduce job

This topic describes how to quickly run a MapReduce program after the MaxCompute client is installed.

Context

A WordCount program is used as an example in this topic.

Before you compile and run a MapReduce program, make sure that the following requirements are met:

- JDK 1.8 has been installed on your host.
- The MaxCompute client has been configured. For more information, see Configure the client.

Procedure

1. Create input and output tables. Example:

   ```
   create table wc_in (key string, value string);
   create table wc_out (key string, cnt bigint);
   ```

   > **Note** For more information about the table creation statement, see Create a table.

2. Use the data transfer tool to upload data. Example:

   ```
   odpscmd -e "tunnel upload kv.txt wc_in"
   ```

3. Compile a MaxCompute program and debug it.
   - MaxCompute provides the Eclipse plug-in to help you quickly develop MapReduce programs and debug them on your local machine.
   - You need to create a MaxCompute project in Eclipse, and then compile a MapReduce program in this project. After successful local debugging, upload the compiled program to

MaxCompute for base testing.

> **Note** If the Java program requires the use of resources, you must use the -resources parameter to specify the resources.

1.4.11. Compile and run a Graph job

You can submit a Graph job in the same way that you would submit a MapReduce job. This topic provides an example of how to submit a Graph job.

Context

This example uses the SSSP algorithm. The operation procedure is as follows:

Procedure

1. Create input and output tables. Example:

```
create table sssp_in (v bigint, es string);
create table sssp_out (v bigint, l bigint);
```

> **Note** For more information about the table creation statement, see Create a table.

2. Use the data transfer tool to upload data. Example:

```
tunnel u -fd " " sssp.txt sssp_in;
```

3. Compile an SSSP.

> **Note** During the Graph development process, you can locally compile and debug SSSP algorithm examples. You only need to package the SSSP code. You do not need to package the SDK into *odps-graph-example-sssp.jar*.

4. Add JAR resources. Example:

```
add jar $LOCAL_JAR_PATH/odps-graph-example-sssp.jar odps-graph-example-sssp.jar
```

> **Note** For more information about how to add resources, see Add resources.

5. Run the SSSP. Example:

```
jar -libjars odps-graph-example-sssp.jar -classpath $LOCAL_JAR_PATH/odps-graph-example-sssp.
jar com.aliyun.odps.graph.examples.SSSP 1 sssp_in sssp_out;
```

> **Note**
>
> The MaxCompute client provides a jar command to run MaxCompute Graph jobs. This command is used in the same way as you would use the jar command in MapReduce. The Graph job execution command outputs the job instance ID, execution progress, and result summary. The command output is as follows:
>
> ID = 20170730160742915gl205u3 2017-07-31 00:18:36 SUCCESS
>
> Summary:

```
Graph Input/Output Total input bytes=211 Total input records=5 Total output bytes=161
Total output records=5 graph_input_[bsp.sssp_in]_bytes=211 graph_input_[bsp.sssp_in]_recor
ds=5 graph_output_[bsp.sssp_out]_bytes=161 graph_output_[bsp.sssp_out]_records=5 Graph
Statistics
Total edges=14
Total halted vertices=5 Total sent messages=28 Total supersteps=4 Total vertices=5
Total workers=1 Graph Timers
Average superstep time (milliseconds)=7 Load time (milliseconds)=8
Max superstep time (milliseconds) =14 Max time superstep=0
Min superstep time (milliseconds)=5 Min time superstep=2
Setup time (milliseconds)=277 Shutdown time (milliseconds)=20
Total superstep time (milliseconds)=30 Total time (milliseconds)=344
OK
```

1.4.12. View job running information

This topic describes how to use the LogView tool to query job running information.

LogView is a tool used to view and debug tasks after a job is submitted to MaxCompute. You can use LogView to view the following content of a job:

- Running status of tasks
- Running results of tasks
- Details of each task and the progress of each step

After a job is submitted to MaxCompute, a link to LogView is generated. You can open the link in a browser to view the job running information.

```
odps@ test_workshop001>select * from result_test;

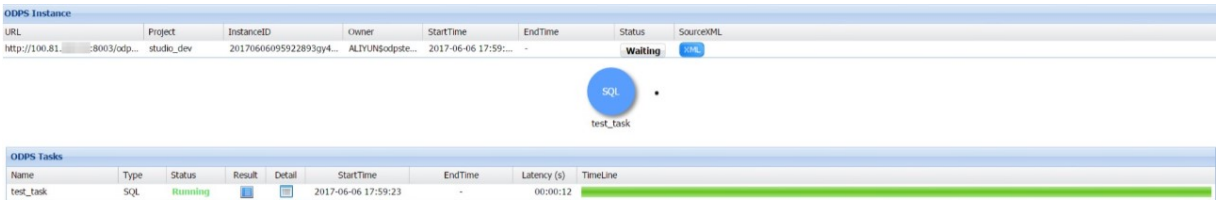
ID = 20190917055240226ga6e5mim
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.cn.maxcomp
m/api&p=test_workshop001&i=20190917055240226ga6e5mim&token=YkxhUzJQ
1NIamRWNDBBU3UJPSxPRFBTX09CTzoxMDc5OTI2ODk2OTk5NDIxLDE1NjkzMDQzNjAs
nQiOlt7IkFjdGlvbiI6WyJvZHBzOlJ1YWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNv
3M6b2RwczoqOnByb2p1Y3RzL3R1c3Rfd29ya3Nob3AwMDEvaW5zdGFuY2UzLzIwMTkw
jI2Z2E2ZTUtaW0iXX1dLCJWZXJzaW9uIjoiMSJ9
Job Queueing...
Summary:

+------------+------------+
| education  | num        |
+------------+------------+
```

Note The LogView page of each job is valid for seven days.

Features

This section describes components on the LogView Web UI.



The LogView page is composed of two parts:

- Instance information
- Task information

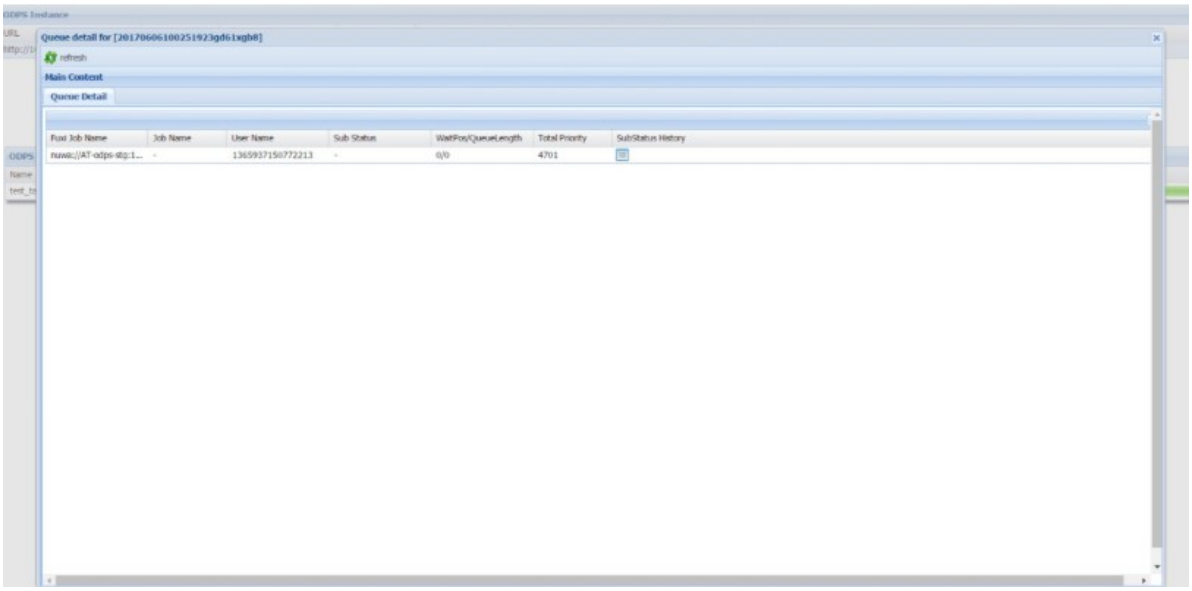
Instance information

The upper pane of the LogView page displays information about the MaxCompute instance corresponding to the SQL tasks that you submit, such as URL, Project, InstanceID, Owner, StartTime, EndTime, and Status.

- If the value of Status is one of the following states, you can click the value to view the queue information:
  - **Waiting**: The job is being processed in MaxCompute and has not been submitted to Job Scheduler.
  - **Waiting List: n**: The job has been submitted to Job Scheduler and is waiting to run in the queue with n-1 other jobs ahead of it.
  - **Running**: The job is running in Job Scheduler.

> **Note** If the value of Status is Terminated, the job has been terminated and has no queue information.

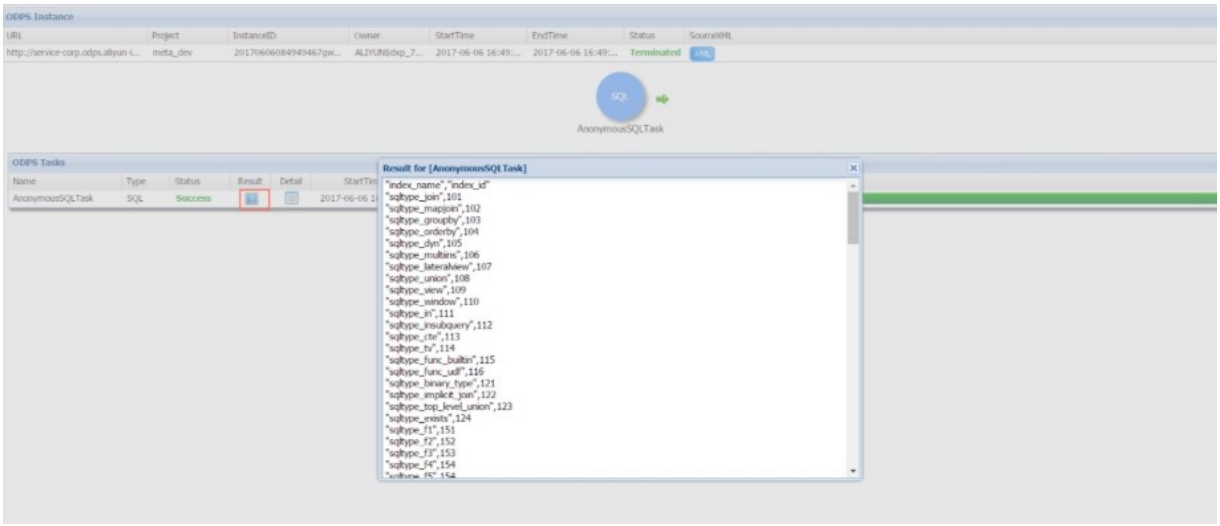
- After you click the value of Status, the following queue information is displayed:
  - **Sub Status**: the current sub-status information.
  - **WaitPos**: the position in the queue. If the value is 0, the job is running. If the value is -, the job has not been submitted to Job Scheduler.
  - **QueueLength**: the total queue length in Job Scheduler.
  - **Total Priority**: the running priority assigned by the system.
  - **SubStatus History**: You can click the icon in this column to view a detailed status history, such as the status code, status description, start time, and duration of a state. This information is unavailable in some versions.



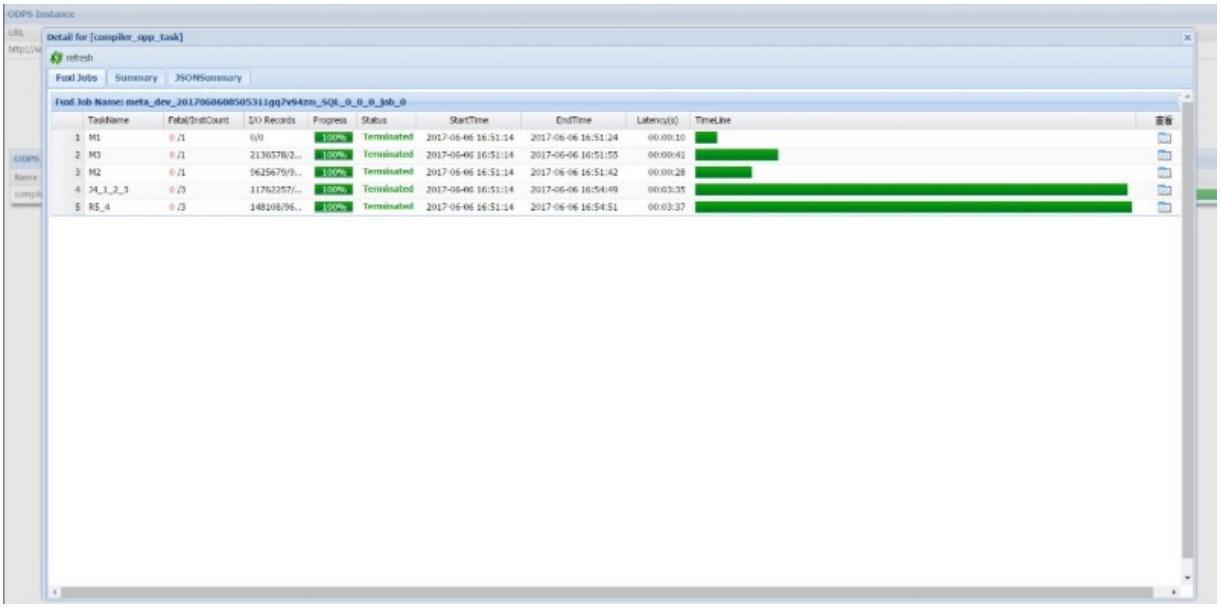
Task information

The lower pane of the LogView page displays the task information corresponding to the instance, such as Name, Type, Status, Result, Detail, StartTime, EndTime, Latency (s), and TimeLine. Like on other pages, latency indicates the total running duration.

Result: You can click the icon in this column to view the task running results after the job is finished. The following figure shows the execution results of a SELECT statement.



Detail: You can click the icon in the **Detail** column to view the task running details regardless of whether the job is running or finished.

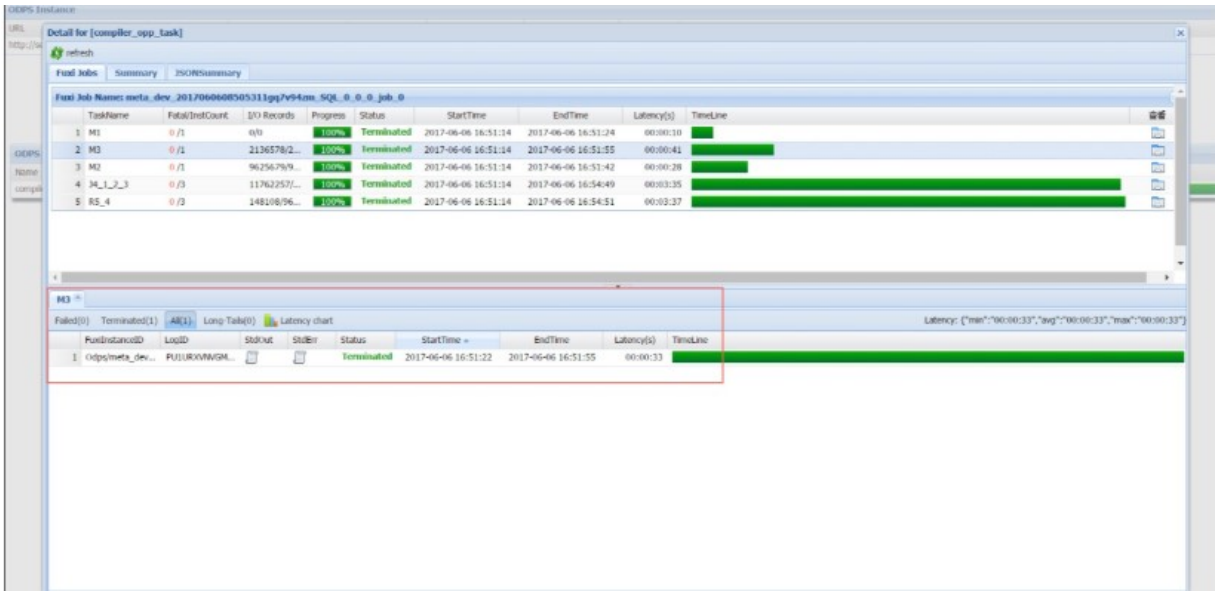


The details about a MaxCompute task are displayed in the detail dialog box. Pay attention to the following points:

- A MaxCompute task consists of one or more Fuxi jobs. For example, if an SQL task that you submit is complex, MaxCompute automatically submits multiple Fuxi jobs to Job Scheduler.
- Each Fuxi job consists of one or more Fuxi tasks. For example, a simple MapReduce task generates two Fuxi tasks: Map Task (M1) and Reduce Task (R2). A complex SQL task may also generate multiple Fuxi tasks.
- The name of a Fuxi task is composed of a letter that indicates the task type and one or more numbers that indicate the task number and its dependencies. For example, in the preceding figure, M1, M2, and M3 are Map tasks, J4_1_2_3 is a Join task that is started after M1, M2, and M3 are finished, and R5_4 is a Reduce task that is started after J4_1_2_3 is finished.
- I/O Records indicates the numbers of input and output records of a Fuxi task.

To view the information about the corresponding instances, you can click the icon in the Show Detail column corresponding to a Fuxi task or double-click the Fuxi task.

> **Note** Each Fuxi task consists of one or more Fuxi instances. As the number of input records of a Fuxi task increases, MaxCompute automatically starts more nodes to help the task process data. Each node corresponds to a Fuxi instance.



In the lower pane of the dialog box, information about the Fuxi instances is displayed in groups. For example, you can click the Failed tab to view the nodes where errors occurred. You can click the icon in the StdOut column to view the standard output information, or click the icon in the StdErr column to view the standard error information.

> **Note** Printed information written in the submitted MaxCompute task is also displayed in the standard output information and standard error information.

Use LogView for troubleshooting

- **Tasks with errors**

  If an error occurs during a task, you can click the icon in the Result column corresponding to the task in the lower pane of the LogView page to view the error information. Alternatively, you can click the icon in the StdErr column corresponding to a Fuxi instance in the detail dialog box to view the error information about the instance.

- **Data skew**

  The long tail of one or more Fuxi instances may slow down the corresponding Fuxi task. The long tail is caused by uneven data distribution in the task. After the task is completed, you can view the running results on the Summary tab of the detail dialog box. The following output provides an example of the running results.

  ```
  output records:
  R2_1_Stg1: 199998999  (min: 22552459, max: 177446540, avg: 99999499)
  ```

  If the difference between the min and max values is large, data skew occurs. For example, if a

specific value appears more often than other values in a column, data skew occurs when you execute a JOIN operation based on this field.

1.5. Basic concepts and common commands

1.5.1. Terms

This topic describes the basic terms of MaxCompute.

Project

A basic organizational unit of MaxCompute. Like a database or schema in a traditional database system, a project is the basic unit of multi-user isolation and access control. A user can have permissions on multiple projects. After security authorization, you can access objects such as tables, resources, functions, and instances in a project from another project.

You can run the Use Project command to access a project. Example:

```
use my_project
-- Access a project named my_project.
```

> **Note** After you run the preceding command, you are navigated to a project named my_project and gain permissions to manage objects in this project. You are then able to manage objects that belong to this project, regardless of which project you are currently in. The Use Project command is provided by the MaxCompute client. For more information, see Common commands.

Table

A data storage unit of MaxCompute. Logically, a table is a two-dimensional structure consisting of rows and columns, in which each row represents a record and each column represents a field of the same data type. One record can contain one or more columns. The column names and types constitute the schema of this table. All data in MaxCompute is stored in tables. Data in table columns can be any of the data types supported by MaxCompute. Tables are the input and output objects of all MaxCompute computing tasks. You can create and delete tables, or import data to tables.

Partitions of a table can be defined to process data more efficiently. You can specify some fields in the table as partition columns. Partitions within a table are similar to directories within a file system. Each value of a partition column is called a partition in MaxCompute. You can group multiple fields of a table to a single partition to create a multi-level partition. Multi-level partitions are similar to multi-level directories. If you specify the name of the partition that you want to access, MaxCompute only scans the specified partition. This improves processing efficiency and reduces cost. For more information, see Column and partition operations.

Data type

Columns of a MaxCompute table must be of a certain data type. MaxCompute supports the following data types:

Basic data types

Type	New in MaxCompute 2.0?	Constant	Description	Value range
TINYINT	Yes	1Y,-127Y	The 8-bit signed integer type.	-128 to 127
SMALLINT	Yes	32767S,-100S	The 16-bit signed integer type.	-32768 to 32767
INT	Yes	1000,-15645787	The 32-bit signed integer type.	-2^{31} to $2^{31}-1$
BIGINT	No	100000000000L,-1L	The 64-bit signed integer type.	$-2^{63}+1$ to $2^{63}-1$
STRING	No	abc, bcd, alibaba, inc	The UTF-8 coded string. The character behaviors of other codes are not defined.	The size of all values in a string column cannot exceed 8 MB.
FLOAT	Yes	None	The 32-bit binary floating point type.	/
BOOLEAN	No	True,False	The Boolean type.	True or False
DOUBLE	No	3.1415926 1E+7	The 64-bit binary floating point type.	$-1.0\ 10^{308}$ to $1.0\ 10^{308}$
DATETIME	No	Datetime '2017-11-11 00:00:00'	The date and time type. The standard system time is UTC+8.	0001-01-01 00:00:00 000 to 9999-12-31 23:59:59 999
DECIMAL	No	3.5BD, 99999999999.9999999BD	The precise numeric type based on the decimal system.	Integer: $-10^{36}+1$ to $10^{36}-1$ Fractional: round to 10^{-18}
VARCHAR	Yes	None	The variable-length type. n specifies the length.	1 to 65535
BINARY	Yes	None	The binary data type.	A single binary column cannot exceed 8 MB.

Type	New in MaxCompute 2.0?	Constant	Description	Value range
TIMESTAMP	Yes	Timestamp '2017-11-11 00:00:00.123456789'	The timestamp type. This type is not time zone specific.	0001-01-01 00:00:00 000000000 to 9999-12-31 23:59:59 999999999

Note that if you want to use the new data types in MaxCompute 2.0, you must first execute the following statement to enable the new data type flag: `set odps.sql.type.system.odps2=true;` (at the session level) or `setproject odps.sql.type.system.odps2=true;` (at the project level). Otherwise, the following error may occur: `xxxx type is not enabled in current mode`. The data types listed in the preceding table can be NULL.

Note Only lowercase letters can be used in the preceding statements.

Notice

Note the following points when you use the new data types in MaxCompute 2.0:

- After you execute the `set odps.sql.type.system.odps2=true;` statement, it results in the following major impacts:
  - The semantics of the INT keyword change. INT in an SQL statement indicates a 32-bit integer.
  - The semantics of an integer constant change. Take the `SELECT 1 + a;` statement as an example.
    - If the new data type flag is not enabled, the integer constant is processed as BIGINT. If the length of the constant exceeds the range for a BIGINT value, the integer constant is processed as DOUBLE.
    - If the new data type flag is enabled, the integer constant is 1 of the 32-bit INT type.
    - Possible compatibility issues: The INT type may lead to inconsistencies in function prototypes during subsequent operations. For example, the actions of peripheral tools and subsequent operations might be changed by new type tables generated after data is written to a disk.
  - Implicit type conversion rules change.

    If the new data type flag is enabled, some implicit types may not be converted. For example, errors may occur or precision may be reduced when the data type is converted from STRING to BIGINT, from STRING to DATETIME, from DOUBLE to BIGINT, from DECIMAL to DOUBLE, or from DECIMAL to BIGINT. In these cases you can use the CAST function to convert the data type.

    Implicit type conversion greatly affects functions and INSERT statements. For example, an INSERT statement can be executed when the new data type flag is disabled, but returns an error when the flag is enabled.

- When the new data type flag is disabled, some operations and built-in functions that use new data types as parameters and response values are ignored. When the new data type flag is enabled, they become valid.
  - Some built-in functions can only be used after the new data type flag is enabled. This includes most functions that use INT type parameters and subsequently suffer from BIGINT overload, such as YEAR, QUARTER, MONTH, DAY, HOUR, MINUTE, SECOND, MILLISECOND, NANOSECOND, DAYOFMONTH, and WEEKOFYEAR. These functions can be implemented by using the DATEPART function.
  - UDF resolution changes.
- The resolution of the BIGINT keyword changes.
- The partition column types change.
  - If the new data type flag is disabled, the partition column type can only be STRING.
  - If the new data type flag is enabled, the partition column type can be STRING, VARCHAR, CHAR, TINYINT, SMALLINT, INT, or BIGINT.
  - If the new data type flag is disabled, partition fields in INSERT operations are processed as STRING.
- The behavior of the LIMIT statement changes.

  Take the `SELECT * FROM t1 UNION ALL SELECT * FROM t2 limit 10;` statement as an example.
  - If the new data type flag is disabled, it is `SELECT * FROM t1 UNION ALL SELECT * FROM ( SELECT * FROM t2 limit 10) t2;` .
  - If the new data type flag is enabled, it is `SELECT * FROM (SELECT * FROM t1 UNION ALL SELECT * FROM t2 ) t limit 10;` .

  Actions of the UNION, INTERSECT, EXCEPT, LIMIT, ORDER BY, DISTRIBUTE BY, SORT BY, and CLUSTER BY statements also change if the new data type flag is enabled.
- The resolution of the IN expression changes.

  Take the a in (1, 2, 3) expression as an example.
  - If the new data type flag is disabled, all the values in the parentheses () must be of the same type.
  - If the new data type flag is enabled, all the values in the parentheses () are implicitly converted to the same type.

- If the value of a constant is greater than the maximum value of INT but less than the maximum value of BIGINT, it is converted to BIGINT. If the constant is greater than the maximum value of BIGINT, it is converted to DOUBLE. If odps.sql.type.system.odps2 is not set to true, MaxCompute retains the conversion and notifies you that the INT data is being processed as the BIGINT type. If odps.sql.type.system.odps2 is not set to true, we recommend that you change these types to BIGINT to prevent confusion.
- VARCHAR constants can be expressed through implicitly converted STRING constants.
- STRING constants can be combined. For example, abc and xyz can be combined as

abcxyz. Different parts can be written in different rows.

- Time values in milliseconds cannot be displayed. You can add `-dfp` in the Tunnel command to specify the time display format in milliseconds.

MaxCompute supports complex data types. The following table lists their definitions and constructors.

Complex data types

Type	Definition	Constructor
Array	array< int >; array< struct< a:int, b:string >>	array(1, 2, 3); array(array(1, 2); array(3, 4))
Map	map< string, string >; map< smallint, array< string>>	map("k1", "v1","k2", "v2"); map(1S, array('a', 'b'), 2S, array('x', 'y'))
Struct	struct< x:int, y:int>; struct< field1:bigint, field2:array< int>, field3:map< int, int>>	named_struct('x', 1, 'y', 2); named_struct('field1', 100L, 'field2', array(1, 2), 'field3', map(1, 100, 2, 200)

If PyODPS is used, you can use the following methods to enable the new data type flag:

- You can execute `o.execute_ SQL ('set ODPS. SQL .type.system.odps2=true;query_ SQL ', hints={" ODPS. SQL .submit.mode": "script"})` to enable the new data type flag.
- You can enable the new data type flag by using DataFrame. For example, use an immediately executed action, such as persist, execute, or to_pandas, by setting the hints parameter. The setting in the following figure is valid only for a single job.

```
from odps.df import DataFrame
users = DataFrame(o.get_table('odps2_test'))
users.persist('copy_test',hints={'odps.sql.type.system.odps2':'true'})
```

- To enable the new data type flag by using DataFrame and have the setting take effect globally, you need to execute `options.sql.use_odps2_extension = True`.

Resource

A concept used in MaxCompute. To accomplish tasks by using user-defined functions (UDFs) or MapReduce, you must use resources.

- MaxCompute SQL UDF: After you write a UDF, you must compile it as a JAR package and upload the package to MaxCompute as a resource. When you run the UDF, MaxCompute automatically downloads the JAR package and obtains the code to run this UDF. JAR packages are a type of

MaxCompute resource. When you upload a JAR package, a resource is created in MaxCompute.

- MaxCompute MapReduce: After you write a MapReduce program, you must compile it as a JAR package and upload the package to MaxCompute as a resource. When you run a MapReduce job, the MapReduce framework automatically downloads the JAR package and obtains the code to run this MapReduce job.

  > Note
  > - There are some limits on how MaxCompute UDFs and MapReduce access resources. For more information, see Limits.
  > - You can also upload tables or text files to MaxCompute as different types of resources. You can read or use these resources when you run UDFs or MapReduce jobs. MaxCompute provides APIs for you to read and use resources. For more information, see the examples for resource use and UDTF instructions.

MaxCompute supports the following resource types:

- File
- Table: tables in MaxCompute.
- JAR: compiled JAR packages.
- Archive: compressed files identified by the resource name extension. Supported file types include .zip, .tgz, .tar.gz, .tar, and .jar.
- Py: Python scripts used by Python UDFs.

> Note For more information about resource operations, see Resource operations.

UDF

MaxCompute provides SQL computing capabilities. You can use the built-in functions in MaxCompute SQL statements to implement certain computing or counting functions. If these built-in functions do not meet your requirements, you can use the Java APIs provided by MaxCompute to develop UDFs. UDFs are classified into user-defined scalar functions (UDSFs), user-defined aggregate functions (UDAFs), and user-defined table-valued functions (UDTFs).

After you write the UDF code, you need to compile the code into a JAR package, upload it to MaxCompute as a resource, and register this UDF in MaxCompute. To use a UDF in MaxCompute, you only need to specify its name and parameters in an SQL statement as you do when you use the built-in functions of MaxCompute.

> Note For more information about UDF operations, see Function operations.

Task

A basic computing unit of MaxCompute. Both SQL and MapReduce features are implemented as tasks. MaxCompute parses most of the tasks that you submit, such as SQL DML statements and MapReduce tasks. MaxCompute generates a task execution plan based on the parsing results.

An execution plan consists of several mutually dependent stages. An execution plan can be logically defined as a directed graph. Vertices of the graph represent stages, and edges of the graph represent dependencies between stages. MaxCompute executes stages based on the dependencies in the graph (execution plan). A stage has multiple processes, also known as workers. The workers in each stage cooperate to complete computations for the stage. Different workers in a stage process different data, but they all use the same execution logic.

A computing task converts to an instance when it is executed. You can perform operations on this instance, such as obtaining status information and killing the instance.

Some MaxCompute tasks, such as SQL DDL statements, are not computing tasks. These tasks only need to read and modify metadata in MaxCompute. MaxCompute does not generate execution plans for these tasks.

> **Notice** MaxCompute does not convert all requests to tasks. For example, project, resource, UDF, and instance operations are not executed as tasks.

Task instance

Some MaxCompute tasks are converted to instances during the execution process. An instance has two stages: Running and Terminated. Instances that are in the Running stage are also in the Running state, while instances that are in the Terminated stage can be in the Success, Failed, or Canceled state. You can query or modify the status of a running task by using the instance ID provided by MaxCompute. Example:

```
status <instance_id>;
-- Query the status of an instance.
kill <instance_id>;
-- Terminate an instance and change its status to Canceled.
```

Resource quota

There are two types of resource quota: storage and computing. The storage quota is the upper limit of storage space configured for a project. If the storage usage approaches the storage quota, an alert is triggered. The computing quota limits the use of memory and CPU resources. The memory and CPUs used to run processes in a project cannot exceed the computing quota.

1.5.2. Common commands

1.5.2.1. Introduction

MaxCompute allows you to perform operations on objects, such as projects, tables, resources, and instances. You can use client commands or the SDK to perform operations on these objects.

This topic describes how to run these commands in the MaxCompute client.

> **Note**
>
> - For more information about how to install and configure the client, see Configure the client.
> - For more information about the SDK, see SDK introduction.

1.5.2.2. Project operations

This topic describes common project commands.

Create or delete a project

MaxCompute does not provide commands for creating or deleting projects. You can configure or operate your projects on the console.

Access a project

Command syntax:

```
use <project_name>;
```

Purpose: It is used to access the specified project. After you enter a project, you can directly operate all objects in the project.

> **Note** If the specified project does not exist or you have not been added to the project, the system returns an exception and exits.

Example:

```
odps@ myproject> use my_project;
-- my_project is a project that you have permission to access.
```

Note

- The preceding command runs in the client.
- All command keywords, project names, table names, and column names in MaxCompute are case-insensitive.
- After you run the command, you can directly access objects in this project. Example:

  Run the following command to access the test_src table in the my_project project (assume test_src exists in my_project):

  ```
  odps@ myproject>select * from test_src;
  ```

  MaxCompute automatically searches for the table from my_project. If this table exists, its data is returned. Otherwise, the system returns an exception and exits.

  If you are in my_project and want to access the test_src table in my_project2, you must specify the project name. Run the following command to access test_src in my_project2:

  ```
  odps@ myproject>select * from my_project2.test_src;
  ```

  Data of test_src in my_project2, not my_project, is returned.

View projects

Command syntax:

```
list projects;
```

Purpose: It is used to display all projects in MaxCompute.

Clear objects from a project

Run the following command to view objects in the recycle bin:

```
show recyclebin;
```

Purpose: It is used to list all objects in the project recycle bin.

Note Only the project owner can run this command.

Run the following command to clear all objects from the project recycle bin:

```
purge all;
```

Purpose: It is used to clear all objects from the project recycle bin to release storage space.

Note Only the project owner can run this command.

Run the following command to clear a table:

```
purge table tblname;
```

Purpose: It is used to clear all objects in a specified table from the recycle bin to release the storage space.

> Note
> - If the specified table exists, the project owner and users who have write permissions on the table can run this command.
> - If the table has been deleted using a DROP command, only the project owner can run this command.

1.5.2.3. Table operations

This topic describes common commands for table operations.

Create a table (CREATE TABLE)

Syntax

```
CREATE TABLE [IF NOT EXISTS] table_name
[(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment]
[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)] [LIFECYCLE days]
[AS select_statement];
CREATE TABLE [IF NOT EXISTS] table_name
LIKE existing_table_name;
```

Description: Creates a table.

Example

```
CREATE TABLE IF NOT EXISTS sale_detail( shop_name STRING,
customer_id STRING, total_price DOUBLE)
PARTITIONED BY (sale_date STRING,region STRING);
-- If a table named sale_detail does not exist, a partitioned table with this name is created.
```

> Note
> - Table names and column names are not case-sensitive.
> - A table name or column name can contain only letters, digits, and underscores (_) and must start with a letter. It cannot exceed 128 bytes in length.
> - A comment must be a valid string within 1,024 bytes. Otherwise, an error is returned.
> - For more information about the command, see Create a table.

Change the owner of a table (CHANGEOWNER)

Syntax

```
ALTER TABLE table_name CHANGEOWNER to new_owner;
```

Description: Changes the owner of a table.

Example

```
ALTER TABLE test1 CHANGEOWNER to 'ALIYUN$xxx@aliyun.com';
-- Change the owner of table test1 to ALIYUN$xxx@aliyun.com.
```

Delete a table (DROP TABLE)

Syntax

```
DROP TABLE [IF EXISTS] table_name;
```

Description: Deletes a table.

> **Note** If a command without the IF EXISTS option is executed and the table does not exist, an exception is returned. If a command with this option is executed, a success is returned regardless of whether the table exists.

Parameters

table_name: the name of the table you want to delete.

Example

```
DROP TABLE sale_detail;
-- If the sale_detail table exists, a success is returned.
DROP TABLE IF EXISTS sale_detail;
-- A success is returned regardless of whether the sale_detail table exists.
```

View table information (DESC)

Syntax

```
DESC <table_name>;
-- View information about a table or view.
DESC extended <table_name>;
-- View information about an external table.
```

Description: Returns the information about a specified table. The returned information includes Owner, Project, CreateTime, LastDDLTime, LastModifiedTime, InternalTable (indicates that the object is a table. The value is always YES), Size (table size in bytes), Native Columns (information about non-partition columns, including field, type, and comment), Partition Columns (information about partition columns, including partition name, type, and comment), and Extended Info (information about the external table, such as StorageHandler and Location).

Parameters

table_name: In the first command, it indicates the name of a table or view. In the second command, it indicates the name of an external table.

Example

```
odps@ project_name>DESC sale_detail;
-- Describe a partitioned table.
+------------------------------------------------------------------------------------+
| Owner: ALIYUN$odpsuser@aliyun.com | Project: test_project |
|TableComment: |
+------------------------------------------------------------------------------------+
|CreateTime: 2017-01-01 17:32:13 |
|LastDDLTime: 2017-01-01 17:57:38 |
|LastModifiedTime: 2017-01-01 18:00:00 |
+------------------------------------------------------------------------------------+
|InternalTable: YES | Size: 0 |
+------------------------------------------------------------------------------------+
|Native Columns: |
+------------------------------------------------------------------------------------+
|Field | Type | Comment |
+------------------------------------------------------------------------------------+
|shop_name | string | |
|customer_id | string | |
|total_price | double | |
+------------------------------------------------------------------------------------+
|Partition Columns: |
+------------------------------------------------------------------------------------+
|sale_date | string | |
|region | string | |
+------------------------------------------------------------------------------------+
```

Note

- The preceding command is executed in the MaxCompute client.
- If the described object is a non-partitioned table, the Partition Columns field is not displayed.
- If the described object is a view, the InternalTable field is replaced with VirtualView whose value is always YES, and the Size field is replaced with ViewText, which defines View, such as select * > from src. For more information about views, see Create a view.

View partition information (DESC)

Syntax

```
desc table_name partition(pt_spec);
```

Description: Views the partition information of a partitioned table.

Example

```
odps@ project_name>desc meta.m_security_users partition (ds='20151010');
+------------------------------------------------------------------------------------+
| PartitionSize: 2109112                                                             |
+------------------------------------------------------------------------------------+
| CreateTime:           2015-10-10 08:48:48                                          |
| LastDDLTime:           2015-10-10 08:48:48                                         |
| LastModifiedTime:       2015-10-11 01:33:35                                        |
+------------------------------------------------------------------------------------+
OK
```

Show tables (SHOW TABLES/SHOW TABLES LIKE)

Syntax

```
SHOW TABLES;
SHOW TABLES like 'chart'
```

Description

- **SHOW TABLES**: shows all tables in the current project.
- **SHOW TABLES like 'chart'**: shows the tables whose name matches chart in the current project. You can use regular expressions in the command to filter tables.

Example

```
odps@ project_name>show tables;
odps@ project_name>show tables like 'ods_brand*';
ALIYUN$odps_user@aliyun.com:table_name
.....
```

Note

- The preceding commands are executed in the MaxCompute client.
- odps_user@aliyun.com indicates the name of the user who creates the table.
- table_name indicates the table name.

Show partitions (SHOW PARTITIONS)

Syntax

```
SHOW PARTITIONS <table_name>;
```

Description: Lists all partitions of a table.

Parameters

table_name: the name of the table that you query. An error is returned if the table does not exist or is a non-partitioned table.

Example

```
SHOW PARTITIONS table_name;
partition_col1=col1_value1/partition_col2=col2_value1
partition_col1=col1_value2/partition_col2=col2_value2
```

Note

- The preceding command is executed in the MaxCompute client.
- partition_col1 and partition_col2 indicate the partition columns of the table.
- col1_value1, col2_value1, col1_value2, and col2_value2 indicate the values in the matching columns.

1.5.2.4. Instance operations

This topic describes common commands for instance operations.

Show instances (SHOW INSTANCES/SHOW P)

Syntax

```
SHOW INSTANCES [FROM startdate TO enddate] [number];
SHOW P [FROM startdate TO enddate] [number];
SHOW INSTANCES [-all];
SHOW P [-all];
SHOW P -p <project name>;
```

Description: Shows information about instances that you have created.

Parameters

- **startdate To enddate**: a period of time. Information about the instances created within the specified period is returned. The dates must be in the format of yyyy-mm-dd. This parameter is optional. If it is not specified, information about instances that you have created in the last three days is returned.
- **number**: the number of instances to be returned. Information about the specified number of instances submitted at the time nearest to the current time is returned in chronological order. If this parameter is not specified, information about all instances that meet the requirements is returned.
- **-all**: Information about instances executed in the current project is returned. By default, a maximum of 50 instances can be returned. You can run the command only when you have the LIST permission on the current project. To return more than 50 instances, you must use the -limit number option. For example, you can run the show p -all -limit 100 command to return information about 100 instances executed in the current project.
- **project name**: the project name. The account that you use must be a member of the project.

Output fields include StartTime (in seconds), RunTime (in seconds), Status (instance status), instanceID, and the SQL statement corresponding to the instance. The following code provides an example of the command output:

```
odps@ $project_name>show p;
StartTime          RunTime  Status   InstanceID              Owner      Query
2015-04-28 13:57:55   1s      Success  20150428xxxxxxxxxxxxxxxxxx   ALIYUN$xxxxx@aliyun-inner.com
select * from tab_pack_priv limit 20;
...    ...    ...    ...    ...    ...
...    ...    ...    ...    ...    ...
```

An instance can be in any of the following states:

- **Running**
- **Success**
- **Waiting**
- **Failed**: The job failed, but data in the target table is not modified.
- **Suspended**
- **Cancelled**

View the status of an instance (STATUS)

Syntax

```
STATUS <instance_id>;
```

Description: Views the status of a specified instance, which can be Success, Failed, Running, or Canceled.

> Note If the instance was not created by you, an exception is returned.

Parameters

instance_id: the unique identifier of an instance. It specifies the instance whose status is queried.

Example

```
odps@ $project_name>status 20131225123xxxxxxxxxxxxxxx;
Success
-- Query the status of the instance whose ID is 20131225123xxxxxxxxxxxxxxx. The query result is Succe
ss.
```

> Note The preceding command is run in the MaxCompute client.

Show running jobs (TOP INSTANCE)

Syntax

```
top instance;
top instance -all;
```

> Note Only the project owner and administrator can use the command.

Description

- top instance: obtains information about running jobs that you submit in the current project. The output fields include InstanceID, Owner, Type, StartTime, Progress, Status, Priority, RuntimeUsage (CPU/MEM), TotalUsage (CPU/MEM), and QueueingInfo (POS/LEN).
- top instance -all: obtains information about all running jobs in the current project. By default, a maximum of 50 jobs are returned. To return more records, use the -limit number option.

Example

```
odps@ $project_name>top instance;
```

> Note The preceding command is run in the MaxCompute client.

Stop an instance (KILL)

Syntax

```
kill <instance_id>;
```

Description: Stops an instance. You can only stop an instance in the Running state. Note that this is an abnormal process. A return from the command only means that the system has received the request. It does not mean that the job has been stopped. Therefore, you must run the STATUS command to view the instance status.

Parameters

instance_id: the unique identifier of an instance. It must be the ID of a running instance. Otherwise, an error is returned.

Example

```
odps@ $project_name>kill 20131225123xxxxxxxxxxxxxxx;
-- Stop the instance whose ID is 20131225123xxxxxxxxxxxxxxx.
```

> **Note** The preceding command is run in the MaxCompute client.

Describe an instance (DESC INSTANCE)

Syntax

```
desc instance <instance_id>;
```

Description: Obtains information about the job corresponding to an instance. The obtained information includes the SQL statement, owner, start time, end time, and status.

Parameters

instance_id: the unique identifier of an instance.

Example

```
odps@ $project_name> desc instance 20150715xxxxxxxxxxxxxxx;
ID                          20150715xxxxxxxxxxxxxxx
Owner                       ALIYUN$XXXXXX@alibaba-inc.com
StartTime                   2015-07-15 18:34:41
EndTime                     2015-07-15 18:34:42
Status                      Terminated
console_select_query_task_1436956481295 Success
Query                       select * from mj_test;
-- Query information about the job corresponding to the instance whose ID is 20150715xxxxxxxxxxxxxxx
xx.
```

> **Note** The preceding command is run in the MaxCompute client.

Obtain task operation log information (WAIT)

Syntax

```
wait instance_id;
```

Description: Obtains task operation log information based on an instance ID. The returned information includes a link to LogView. You can view log details by using the link.

Parameters

instance_id: the unique identifier of an instance.

Example

```
wait 20170925161122379gxxxxxx;
ID = 20170925161122379g3xxxxxx
Log view:
http://logview.odps.aliyun.com/logview/?h=http://service.odps.aliyun.com/api&p=alian&i=2017092516
11223xxxxxxdqp&token=XXXXXXvbiI6IjEifQ==
Job Queueing...
Summary:
resource cost: cpu 0.05 Core * Min, memory 0.05 GB * Min
inputs:
    alian.bank_data: 41187 (588232 bytes)
outputs:
    alian.result_table: 8 (640 bytes)
Job run time: 2.000
Job run mode: service job
Job run engine: execution engine
M1:
    instance count: 1
    run time: 1.000
    instance time:
        min: 1.000, max: 1.000, avg: 1.000
    input records:
        TableScan_REL5213301: 41187  (min: 41187, max: 41187, avg: 41187
)
    output records:
        StreamLineWrite_REL5213305: 8  (min: 8, max: 8, avg: 8)
R2_1:
    instance count: 1
    run time: 2.000
    instance time:
        min: 2.000, max: 2.000, avg: 2.000
    input records:
        StreamLineRead_REL5213306: 8  (min: 8, max: 8, avg: 8)
    output records:
        TableSink_REL5213309: 8  (min: 8, max: 8, avg: 8)
-- Query the task operation logs of the instance whose ID is 20170925161122379gxxxxxx.
```

Note The preceding command is run in the MaxCompute client.

1.5.2.5. Resource operations

This topic describes common commands for resource operations.

Add a resource (ADD FILE/ARCHIVE/TABLE/JAR/PY)

Syntax

```
add file <local_file> [as alias] [comment 'cmt'][-f];
add archive <local_file> [as alias] [comment 'cmt'][-f];
add table <table_name> [partition <(spec)>] [as alias] [comment 'cmt'][-f];
add jar <local_file.jar> [comment 'cmt'][-f];
add py <local_file.py> [comment 'cmt'][-f];
```

Description: Adds resources to MaxCompute.

The following table lists parameters in this command.

Parameters

Parameter	Description
file/archive/table/jar/py	Indicates the resource type. For more information about resource types, see Resource in Terms.
local_file	Indicates the path of a local file. The file name is used as the resource name, which uniquely identifies the resource.
table_name	Indicates the name of a table in MaxCompute.
[PARTITION (spec)]	If the resource that is added is a partitioned table, MaxCompute only takes a partition as a resource, as opposed to the whole partitioned table.
alias	Indicates the resource name. If this parameter is not specified, the file name is used as the resource name. JAR and Py resources do not support this parameter.
[comment 'cmt']	Indicates a comment on the resource.
[-f]	If a resource with the same name exists, this operation overwrites the existing resource. If this option is not specified and a resource with the same name exists, the operation fails.

Example

```
odps@ odps_public_dev>add table sale_detail partition (ds='20170602') as sale.res comment 'sale deta
il on 201706 02' -f;
OK: Resource 'sale.res' have been updated.
-- Add a table resource with alias sale.res to MaxCompute.
```

> **Note** The size of each resource file cannot exceed 500 MB. The total size of resources referenced by a single SQL or MapReduce task cannot exceed 2,048 MB.

Delete a resource (DROP RESOURCE)

Syntax

```
DROP RESOURCE <resource_name>;
```

Description: Deletes an existing resource.

Parameters

resource_name: the name of the resource to be deleted. It is specified when the resource is created.

View the resource list (LIST RESOURCES)

Syntax

```
LIST RESOURCES;
```

Description: Lists all resources in the current project.

Example

```
odps@ $project_name>list resources;
Resource Name     Comment     Last Modified Time      Type
1234.txt                  2014-02-27 07:07:56     file
mapred.jar                2014-02-27 07:07:57     jar
```

Download a resource (GET RESOURCE)

Syntax

```
GET RESOURCE <resource_name> <path>;
```

Description: Downloads a resource from MaxCompute to a local device. You can download file, JAR, archive, and Py resources, but not table resources.

Parameters

- **resource_name**: the name of the resource to be downloaded.
- **path**: the local path to save the resource.

Example

```
odps@ $project_name>get resource odps-udf-examples.jar d:\;
OK
```

1.5.2.6. Function operations

This topic describes common commands for function operations.

Register a function (CREATE FUNCTION)

Syntax

```
CREATE FUNCTION <function_name> AS <package_to_class> USING<resource_list>;
```

The following table lists parameters in this command.

Parameters

Parameter	Description
function_name	Indicates the UDF name. This name is used in SQL statements to reference this function.
package_to_class	For a Java UDF, package_to_class indicates a fully qualified class name. It consists of names from the top-level package name all the way to the UDF implementation class name, which are separated with periods (.). For a Python UDF, package_to_class indicates the Python script name and class name, which are separated with periods (.). This name must be enclosed in a pair of single quotation marks (').
resource_list	Indicates the list of resources that the UDF uses. The list must include the resource where the UDF code is located. Ensure that the resource is uploaded to MaxCompute before you register the function. If the user code reads resource files by using the distributed cache API, this list must also include all resource files that are read by the UDF. If the resource list contains multiple resources, separate them with commas (,). The resource list must be enclosed in a pair of single quotation marks ('). If you need to specify the project where the resource is located, use the <project_name>/resources/<resource_name> format.

Example

- Assume that Java UDF class org.alidata.odps.udf.examples.Lower is in my_lower.jar. Execute the following statement to create the my_lower function:

```
CREATE FUNCTION my_lower AS 'org.alidata.odps.udf.examples.Lower'
USING 'my_lower.jar';
```

- Assume that Python UDF class MyLower is in the pyudf_test.py script of the test_project project. Execute the following statement to create the my_lower function:

```
create function my_lower as 'pyudf_test.MyLower'
using 'pyudf_test.py';
```

- Assume that Java UDF class com.aliyun.odps.examples.udf.UDTFResource is in udtfexample1.jar and the function depends on file resource file_resource.txt, table resource table_resource1, and archive resource test_archive.zip. Execute the following statement to create the test_udtf function:

```
create function test_udtf as 'com.aliyun.odps.examples.udf.UDTFResource'
using 'udtfexample1.jar, file_resource.txt, table_resource1,test_archive.zip';
```

> Note
> - Similar to resource file names, function names must be unique.
> - Only the project owner has the permission to use UDFs to overwrite built-in functions. If you use a UDF that overwrites a built-in function, warning information is included in Summary after the SQL statement is executed.

Delete a function (DROP FUNCTION)

Syntax

```
DROP FUNCTION <function_name>;
```

Parameters

function_name: the name of the function you want to delete.

Example

```
DROP FUNCTION test_lower;
```

View the function list (LIST FUNCTIONS)

Syntax

```
LIST FUNCTIONS;
-- View all UDFs in the current project.
LIST FUNCTIONS -p project_name;
-- View all UDFs in a specified project.
```

1.5.2.7. Time zone configuration operations

This topic describes how to use a SET command to configure the time zone of a MaxCompute project.

Time zone configurations

By default, the time zone of a MaxCompute project is UTC+8. The DATETIME, TIMESTAMP, and DATE fields and the related built-in time functions are all calculated based on UTC+8. You can use one of the following methods to configure the time zone:

- **Session level:** Submit the set odps.sql.timezone=<timezoneid>; statement along with a computing statement for execution. Example:

```
set odps.sql.timezone=Asia/Tokyo;
-- Set the time zone to Asia/Tokyo.
select getdate();
-- Query the current time zone.
output:
+------------+
| _c0        |
+------------+
| 2018-10-30 23:49:50 |
+------------+
```

- **Project level:** Execute the setProject odps.sql.timezone=<timezoneid>; statement. Only the project owner has the permission to execute this statement.

> **Notice** After the time zone is set at the project level, it is used in all computing tasks that are time related. The data of existing tasks are also affected. Therefore, set the time zone only when it is absolutely necessary. We recommend that you only set time zones for new projects.

Limits and precautions

- SQL built-in date functions, UDFs, UDTs, UDJs, and the SELECT TRANSFORM statement all support time zone configuration at the project level.
- A time zone must be configured in the format such as Asia/Shanghai, which supports daylight saving time. Do not configure it in the GMT+9 format.
- If the time zone of the SDK differs from that of the project, you must configure the GMT time zone to convert the data type from DATETIME to STRING.
- After you configure the time zone, the output time may be different from the real time when you run the related SQL statements by using DataWorks. For data that is between the years 1900 and 1928, the time difference is 352 seconds. For data that is before the year 1900, the time difference is 9 seconds.
- New versions of MaxCompute, Java SDK, and the related console are provided to ensure that the DATETIME data stored in MaxCompute is correct in multiple time zones. The target Java SDK and console versions have the -oversea suffix. The upgrade may affect the display of the DATETIME data that is before January 1, 1928 in MaxCompute.
- If the local time zone is not UTC+8, we recommend that you upgrade the Java SDK and console to ensure that the SQL-based computing results and data transferred using Tunnel after January 1, 1900 are accurate and consistent. After the upgrade, for the DATETIME data before January 1, 1900, the SQL-based computing results and data transferred using Tunnel may still have a difference of 343 seconds. For the DATETIME data between January 1, 1900 and January 1, 1928 uploaded before the upgrade, the time in the new version is 352 seconds earlier.
- If you do not upgrade the Java SDK and console to the version with the -oversea suffix, the SQL-based computing results and data transferred using Tunnel will have differences. For data before January 1, 1900, the time difference is 9 seconds. For data between January 1, 1900

and January 1, 1928, the time difference is 352 seconds.

> **Note** The time zone of the upgraded Java SDK or console version does not affect the time zone configuration in DataWorks. Therefore, the time zones may vary. You must evaluate the impact on scheduled tasks in DataWorks.

- If you use a third-party client that is connected to MaxCompute by using JDBC, you must set the time zone on the client to ensure the time consistency between the client and the server.
- MapReduce supports time zone configuration.
- Spark supports time zone configuration.
  - If tasks are submitted to the MaxCompute computing cluster, the time zone of the project can be automatically obtained.
  - If tasks are submitted from spark-shell, spark-sql, or pyspark in yarn-client mode, you must configure the spark-defaults.conf parameter of the driver and add the spark.driver.extraJavaOptions -Duser.timezone=America/Los_Angeles statement. The timezone parameter in the preceding statement indicates the time zone to be used.
- PAI supports time zone configuration.
- Graph supports time zone configuration.

1.5.2.8. Tunnel operations

This topic describes common commands for Tunnel operations.

Features

- **UPLOAD**: uploads files or directories (level-1 directories). Data can be uploaded only to a table or a partition of a table. For a partitioned table, you must specify the partition where data is upload.

```
tunnel upload log.txt test_project.test_table/p1="b1",p2="b2";
tunnel upload log.txt test_table --scan=only;
```

- **DOWNLOAD**: downloads a table or partition to a single file. For a partitioned table, specify the partition to be downloaded.

```
tunnel download test_project.test_table/p1="b1",p2="b2" log.txt;
```

- **RESUME**: resumes the transfer of files or directories after a network or Tunnel service error occurs. Dship supports resumable data transfer.

```
tunnel resume;
```

- **SHOW**: displays historical task information.

```
tunnel show history -n 5
tunnel show log
```

- **PURGE**: clears the session directory. Sessions within the past three days are purged by default.

```
tunnel purge 5
```

Use of Tunnel commands

Tunnel commands allows you to obtain help information by using the Help subcommand on the client. Each command and option support short command format:

```
odps@ project_name>tunnel help;
Usage: tunnel <subcommand> [options] [args]
Type 'tunnel help <subcommand>' for help on a specific subcommand.

Available subcommands:
   upload (u)
   download (d)
   resume (r)
   show (s)
   purge (p)
   help (h)
tunnel is a command for uploading data to / downloading data from ODPS.
```

The following table describes parameters in this command.

Parameters

Parameter	Description
upload	Uploads data to MaxCompute tables.
download	Downloads data from MaxCompute tables.
resume	Resumes data upload if a failure occurs. Currently, data download cannot be resumed. Each data upload or download operation is called a session. To resume a session, you must specify the session ID in the RESUME command.
show	Displays historical running information.
purge	Clears the session directory.
help	Provides Tunnel help information.

UPLOAD

Imports data from a local file to a MaxCompute table in append mode. Subcommand syntax:

```
usage: tunnel upload [options] <path> <[project.]table[/partition]>
            upload data from local file
 -bs,-block-size <ARG>            block size in MiB, default 100
 -c,-charset <ARG>                specify file charset, default ignore.
                                  set ignore to download raw data
 -cp,-compress <ARG>              compress, default true
 -dbr,-discard-bad-records <ARG>  specify discard bad records
                                  action(true|false), default false
 -dfp,-date-format-pattern <ARG>  specify date format pattern, default
                                  yyyy-MM-dd HH:mm:ss
 -fd,-field-delimiter <ARG>       specify field delimiter, support
                                  unicode, eg \u0001. default ","
 -h,-header <ARG>                 if local file should have table header,
                                  default false
 -mbr,-max-bad-records <ARG>      max bad records, default 1000
 -ni,-null-indicator <ARG>        specify null indicator string, default
                                  ""(empty string)
 -rd,-record-delimiter <ARG>      specify record delimiter, support
                                  unicode, eg \u0001. default "\n"
 -s,-scan <ARG>                   specify scan file
                                  action(true|false|only), default true
 -sd,-session-dir <ARG>           set session dir, default /D:/console/plugins/dship/
 -te,-tunnel_endpoint <ARG>       tunnel endpoint
   -threads <ARG>                 number of threads, default 1
 -tz,-time-zone <ARG>             time zone, default local timezone:
                                  Asia/Shanghai
Example:
   tunnel upload log.txt test_project.test_table/p1="b1",p2="b2"
```

The following table describes parameters in this command.

Parameters

Parameter	Description
-bs,-block-size	Specifies the size of each data block uploaded using Tunnel. Default value: 100 MiB (1 MiB = 1,024 × 1,024 bytes).
-c,-charset	Specifies the encoding format of local data files. The default value is UTF-8 without timing. The source data is downloaded by default.

Parameter	Description
-cp,-compress	Specifies whether to compress the local file before uploading it to reduce traffic. Compression is enabled by default.
-dbr	Specifies whether to ignore dirty data, such as additional columns, missing columns, and unmatched types of column data. If the value is true, all data that does not comply with table definitions is ignored. If the value is false, an error is returned when dirty data is found, and raw data in the destination table is not contaminated.
-dfp	Specifies the format of DATETIME data. The default format is yyyy-MM-dd HH:mm:ss.
-fd	Specifies the column delimiter used in the local data file. The default delimiter is comma (,).
-h	Specifies whether the data file has a table header. If the value is true, Dship skips the table header and starts uploading data from the second row.
-mbr,-max-bad-records	If more than 1,000 rows of dirty data are uploaded, the upload operation is terminated by default. This parameter allows you to adjust the maximum allowable volume of dirty data.
-ni	Specifies the NULL data identifier. The default value is an empty string ("").
-rd	Specifies the row delimiter in the local data file. The default value is \n in a Linux system or \r\n in a Windows system.
-s	Specifies whether to scan the local data file. The default value is false. If the value is true, data is scanned first, and is imported if the format is correct. If the value is false, data is imported without scanning. If the value is only, data is scanned but not imported.
-sd,-session-dir	Specifies the path of the session directory. The default value is */D:/console/plugins/dship/lib/...*
-te	Specifies the endpoint of Tunnel.
-tz	Specifies the time zone. The default value is Asia/Shanghai.

Example

Create a table:

```
CREATE TABLE IF NOT EXISTS sale_detail(
   shop_name STRING,
   customer_id STRING,
    total_price DOUBLE)
PARTITIONED BY (sale_date STRING,region STRING);
```

Add a partition:

```
alter table sale_detail add partition (sale_date='201705', region='hangzhou');
```

Prepare the data file data.txt with the following content:

```
shop9,97,100
shop10,10,200
shop11,11
```

The third row of this file does not comply with the definitions of the sale_detail table. sale_detail defines three columns, but the third row of this file contains only two columns. Run the following command to import the data:

```
odps@ project_name>tunnel u d:\data.txt    sale_detail/sale_date=201705,region=hangzhou -s false U
pload session:    20170610163922488087 0a002ec60c
Start upload:d:\data.txt
Total bytes:41 Split input to 1 blocks
2017-06-10 16:39:22 upload block: '1'
ERROR: column mismatch -,expected 3 columns, 2 columns found, please check    data or delimiter
```

The data import fails because of the dirty data in data.txt. The session ID and error message are displayed. You can run the following command to verify the upload result:

```
odps@ odpstest_ay52c_ay52> select * from sale_detail where sale_date='201705'; ID = 20170610084135
370gyvc61z5
+-----------+-------------+-------------+-----------+--------+
shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
+-----------+-------------+-------------+-----------+--------+
```

DOWNLOAD

Subcommand syntax

```
odps@ project_name>tunnel help download
usage: tunnel download [options] <[project.]table[/partition]> <path>
          download data to local file
 -c,-charset <ARG>              specify file charset, default ignore.
                          set ignore to download raw data
 -cp,-compress <ARG>            compress, default true
 -dfp,-date-format-pattern <ARG>   specify date format pattern, default
                          yyyy-MM-dd HH:mm:ss
 -e,-exponential <ARG>          When download double values, use
                          exponential express if necessary.
                          Otherwise at most 20 digits will be
                          reserved. Default false
 -fd,-field-delimiter <ARG>       specify field delimiter, support
                          unicode, eg \u0001. default ","
 -h,-header <ARG>               if local file should have table header,
                          default false
   -limit <ARG>                 specify the number of records to
                          download
 -ni,-null-indicator <ARG>        specify null indicator string, default
                          ""(empty string)
 -rd,-record-delimiter <ARG>      specify record delimiter, support
                          unicode, eg \u0001. default "\n"
 -sd,-session-dir <ARG>         set session dir, defa          /D:/console/plugins/dship/
-te,-tunnel_endpoint <ARG>       tunnel endpoint
   -threads <ARG>               number of threads, default 1
 -tz,-time-zone <ARG>           time zone, default local timezone:Asia/Shanghai
Example:
   tunnel download test_project.test_table/p1="b1",p2="b2" log.txt
```

The following table describes parameters in this command.

Parameters

Parameter	Description
-fd	Specifies the column delimiter used in the local data file. The default delimiter is a comma (,).
-rd	Specifies the row delimiter used in the local data file. The default delimiter is \r\n.
-dfp	Specifies the format of DATETIME data. The default format is yyyy-MM-dd HH:mm:ss.

Parameter	Description
-ni	Specifies the NULL data identifier. The default value is an empty string ("").
-c	Specifies the encoding format of local data files. The default value is UTF-8.

Example

Download data to result.txt:

```
$ ./tunnel download sale_detail/sale_date=201705,region=hangzhou result.txt; Download session: 20
170610165824528387 0a002ed0b9
Total records: 2
2017-06-10 16:58:24 download records: 2
2017-06-10 16:58:24 file size: 30 bytes
OK
```

Open the result.txt file and verify that its content is as follows:

```
shop9,97,100.0
shop10,10,200.0
```

RESUME

Resumes the execution of historical operations. Only data upload can be resumed. Subcommand syntax:

```
usage: tunnel resume [session_id] [--force]
            resume an upload session
 -f,-force   force resume
Example:
   tunnel resume
```

Example

Change the content of the data.txt file to the following items:

```
shop9,97,100
shop10,10,200
```

Resume the data upload:

```
odps@ project_name>tunnel resume 20170610163922488 0870a002ec60c --      force;
start resume
20170610163922488 0870a002ec60c
Upload session: 20170610163922488 0870a002ec60c
Start upload:d:\data.txt
Resume 1 blocks
2017-06-10 16:46:42 upload block: '1'
2017-06-10 16:46:42 upload block complete, blockid=1
upload complete, average > speed is 0 KB/s
OK
```

20170610163922488 0870a002ec60c is the ID of the session that failed to be uploaded. You can run the following command to verify the upload result:

```
odps@ project_name>select * from sale_detail where sale_date='201705';
ID = 20170610084801405g0a741z5
+-----------+-------------+-------------+-----------+--------+
shop_name | customer_id | total_price | sale_date | region |
+-----------+-------------+-------------+-----------+--------+
+-----------+-------------+-------------+-----------+--------+
```

SHOW

Displays historical records. Subcommand syntax:

```
usage: tunnel show history [options]
```

Example

```
odps@ project_name>tunnel show history;
   usage: tunnel show history [options]
          show session information
 -n,-number <ARG>   lines
Example:
   tunnel show history -n 5
   tunnel show log
```

PURGE

Clears the session directory. Sessions within the last three days are purged by default. Subcommand syntax:

```
usage: tunnel purge [n]
        force session history to be purged.([n] days before, default
        3 days)

Example:
  tunnel purge 5
```

1.5.2.9. Other operations

This topic describes common commands for other operations.

ALIAS

The ALIAS command is used to create an alias for a resource. You can read different resources from the MaxCompute MapReduce reference manual or UDF code by creating the same alias for these resources.

Syntax

```
ALIAS <alias>=<real>;
```

Description: Creates an alias for a resource.

Parameters

- **alias**: the alias of a resource.
- **real**: the original name of a resource.

Example

```
ADD TABLE src_part PARTITION (ds='20171208') AS res_20171208;
ADD TABLE src_part PARTITION (ds='20171209') AS res_20171209;
-- Add resources res_20171208 and res_20171209.
ALIAS resName=res_20171208;
jar -resources resName -libjars work.jar -classpath ./work.jar com.company.MainClass args ...;
-- Set the alias of resource res_20171208 to resName and call this resource.
ALIAS resName=res_20171209;
jar -resources resName -libjars work.jar -classpath ./work.jar com.company.MainClass args ...;
-- Set the alias of resource res_20171209 to resName and call this resource.
```

> **Note** In the preceding example, different resource tables are referenced by the same resource alias resName in two jobs. Different data is read with the same code.

SET

Syntax

```
set <KEY>=<VALUE>
```

Description: Configures built-in or user-defined system variables of MaxCompute to affect MaxCompute behavior.

Parameters

- **KEY**: the name of the attribute to be set.
- **VALUE**: the value of the attribute.

MaxCompute SQL and the latest MapReduce version support the following SET commands:

```
set odps.stage.mapper.mem=
-- Set the memory size of each map worker. Unit: MB. Default value: 1024.
set odps.stage.reducer.mem=
-- Set the memory size of each reduce worker. Unit: MB. Default value: 1024.
set odps.stage.joiner.mem=
-- Set the memory size of each join worker. Unit: MB. Default value: 1024.
set odps.stage.mem =
-- Set the memory size of all workers of a specified MaxCompute task. This command has a lower priori
ty than the preceding commands. Unit: MB. This parameter is not specified by default.
set odps.stage.mapper.split.size=
-- Set the input data volume of each map worker (size of each slice in the input file) to indirectly contr
ol the number of workers in each map stage. Unit: MB. Default value: 256.
set odps.stage.reducer.num=
-- Set the number of workers in each reduce stage. This parameter is not specified by default.
set odps.stage.joiner.num=
-- Set the number of workers in each join stage. This parameter is not specified by default.
set odps.stage.num=
-- Set the number of concurrent workers in all stages of a specified MaxCompute task. This command h
as a lower priority than the preceding three commands. This parameter is not specified by default.
```

The earlier MapReduce versions of MaxCompute support the following SET commands:

```
set odps.mapred.map.memory=
-- Set the memory size of each map worker. Unit: MB. Default value: 1024.
set odps.mapred.reduce.memory=
-- Set the memory size of each reduce worker. Unit: MB. Default value: 1024.
setodps.mapred.map.split.size=
-- Set the input data volume of each map worker (size of each slice in the input file) to indirectly contr
ol the number of workers in each map stage. Unit: MB. Default value: 256.
set odps.mapred.reduce.tasks=
-- Set the number of workers in each reduce stage. This parameter is not specified by default.
```

Example

- Adjust the cache size pre-defined for a complex column when data is written to MaxCompute tables to improve write performance.

```
set odps.sql.executionengine.coldata.deep.buffer.size.max=1048576;
```

- Set the input data volume of each map worker to 256 MB.

```
set odps.stage.mapper.split.size=256
```

SETPROJECT

Syntax

```
setproject ["<KEY>=<VALUE>"];
```

Description: Sets project attributes. If < KEY >=< VALUE > is not specified, the current attribute configurations of the project are displayed.

The following table describes project attributes.

Project attributes

Attribute	Permission owner	Description	Value range
odps.table.drop.ignoreonexistent	All users	Indicates whether to report an error when you try to delete a table that does not exist. If the value is true, no error is reported.	true and false
odps.instance.priority.autoadjust	Project owner	Indicates whether to automatically prioritize smaller tasks. If the value is true, this function is enabled.	true and false
odps.instance.priority.level	Project owner	Indicates the priority of a task in a project.	1 to 3 Note The value 1 indicates the highest priority.
odps.security.ip.whitelist	Project owner	Indicates an IP address whitelist for the project.	List of IP addresses separated with commas (,).

Attribute	Permission owner	Description	Value range
odps.table.lifecycle	Project owner	**optional**: The lifecycle clause is optional in a table creation statement. If no lifecycle is set for a table, the table does not expire. **mandatory**: The lifecycle clause is mandatory. **inherit**: If no lifecycle is set for a table, the value of odps.table.lifecycle.value is the lifecycle of this table.	optional, mandatory, inherit
odps.table.lifecycle.value	Project owner	Indicates the default lifecycle.	1 to 37231 Note The default value is 37231.
odps.instance.remain.days	Project owner	Indicates how long the information about an instance is retained. Unit: days.	3 to 30
odps.task.sql.outerjoin.ppd	Project owner	Indicates whether the filter conditions in FULL OUTER JOIN are pushed down.	true and false
odps.function.strictmode	Project owner	Indicates whether to return NULL (false) or report an error (true) when built-in functions have dirty data.	true and false
odps.task.sql.write.str2null	Project owner	Indicates whether to consider empty strings as NULL (true).	true and false

EXPORT project meta

Syntax

```
export <projectname> <local_path>;
```

Description: Exports the meta of a project to a local file. Meta is represented by statements acceptable by odpscmd. The exported meta file can be used to recreate a project.

SHOW FLAGS

Syntax

```
show flags;
```

Description: Displays parameters configured by using the SET command.

> Note This command takes effect only at the project level.

COST SQL

Syntax

```
cost sql <SQL Sentence>;
```

Description: Estimates the measurement information of an SQL statement, including the size of the input data, number of UDFs, and SQL complexity level.

Example

```
odps@ $odps_project >cost sql select distinct project_name, user_name from meta.m_security_users di
stribute by project_name sort by project_name;

ID = 20190715113033121xxxxxxx
Input:65727592 Bytes
UDF:0
Complexity:1.0
```

1.6. MaxCompute SQL

1.6.1. Overview

1.6.1.1. Scenarios

This topic describes the scenarios of MaxCompute SQL.

MaxCompute SQL offline computing is applicable to scenarios where large volumes of data (terabytes) need to be processed, but do not have high real-time requirements. In such scenarios, it takes a relatively long time to prepare and submit each job. MaxCompute SQL is not well-suited for businesses that require to process thousands of transactions per second. MaxCompute SQL online computing provides near real-time (NRT) processing capabilities.

MaxCompute SQL uses the syntax that is similar to SQL syntax. It can be considered as a subset of standard SQL. However, MaxCompute SQL is not equivalent to a database. It does not have common database characteristics, such as transactions, primary key constrains, and indexes. The maximum length of SQL statements currently supported by MaxCompute is 2 MB.

1.6.1.2. Reserved words

Keywords of SQL statements are reserved words in MaxCompute. Do not use reserved words to name tables, columns, or partitions. Otherwise, an error is returned. Reserved words are case-insensitive.

Common reserved words are listed as follows. For a complete list of reserved words, see Reserved words.

```
% & && ( ) * + - . / ; < <= <>
= > >= ? ADD ALL ALTER
AND AS ASC BETWEEN BIGINT BOOLEAN BY
CASE CAST COLUMN COMMENT CREATE DESC DISTINCT
DISTRIBUTE DOUBLE DROP ELSE FALSE FROM FULL
GROUP IF IN INSERT INTO IS JOIN
LEFT LIFECYCLE LIKE LIMIT MAPJOIN NOT NULL
ON OR ORDER OUTER OVERWRITE PARTITION RENAME
REPLACE RIGHT RLIKE SELECT SORT STRING TABLE
THEN TOUCH TRUE UNION VIEW WHEN WHERE
```

1.6.1.3. Partitioned table

Partition columns provide many benefits, such as higher SQL operating efficiency and lower costs. However, too many partitions can cause problems. Using partition columns as filtering conditions in WHERE clauses of SELECT statements can bring greater benefits. Some SQL partition statements run inefficiently. For example, a statement fails when a large volume of data (more than 2,048 MB) is generated in dynamic partitions in a single MaxCompute instance.

It is easy to underestimate the number of partitions generated when multi-level partitions are used. When a huge number of partitions are generated, you must evaluate the original data to determine if there are excessive partitions.

You can create up to six levels of partitions. For some MaxCompute commands, the syntax differs between partitioned and non-partitioned tables. For more information, see DDL statements and DML statements.

For more information about the table creation statement, see Create a table.

1.6.1.4. Type conversion

1.6.1.4.1. Explicit type conversion

Explicit conversion uses CAST to convert a value type to another one. This topic describes explicit type conversion.

The following table lists explicit type conversions supported by MaxCompute SQL.

Explicit type conversion

From/To	Bigint	Double	String	Datetime	Boolean	Decimal
Bigint	-	Y	Y	N	N	Y
Double	Y	-	Y	N	N	Y
String	Y	Y	-	Y	N	Y
Datetime	N	N	Y	-	N	N
Boolean	N	N	N	N	-	N
Decimal	Y	Y	Y	N	N	-

Y indicates that the type can be converted. N indicates that the type cannot be converted.

Note

- When double type values are converted to bigint, the fractional is truncated. For example, cast(1.6 as bigint) = 1.
- When a string that meets double type requirements is converted to bigint, the string is first converted to the double type before it is converted to the bigint type. Hence, the fractional is truncated. For example, cast("1.6" as bigint) = 1.
- When a string that meets bigint type requirements is converted to the double type, one decimal is retained. For example, cast("1" as double) = 1.0.
- To convert a constant string to the decimal type, enclose the constant string within a pair of quotation marks. If the value is not enclosed in quotation marks, it is treated as a double type value. For example, cast("1.234567890123456789" as decimal).
- Unsupported explicit type conversion operations cause an exception.
- If a conversion fails during execution, the system returns an error and exits.
- The datetime data conversion uses the default format yyyy-mm-dd hh:mi:ss. For more information, see Convert data between string and datetime types.
- Some types cannot be explicitly converted, but can be converted using built-in SQL functions. For example, the to_char function can be used to convert boolean type values to the string type. For more information, see TO_CHAR. The to_date function can be used to convert string type values to the datetime type. For more information, see TO_DATE.
- For more information about CAST, see CAST.
- When the values of the decimal type are out of the value range, the cast string to decimal operation may return an error, such as most significant bit overflow or least significant bit overflow truncation.

1.6.1.4.2. Implicit type conversion and its scope

Implicit type conversion is an automatic type conversion performed by MaxCompute based on the context and a predefined set of rules. This topic describes the rules of implicit type conversion.

The following table lists implicit type conversion rules supported by MaxCompute.

Implicit type conversion 1

From/To	BOOLEAN	TINYINT	SMALLINT	INT	BIGINT	FLOAT
BOOLEAN	T	F	F	F	F	F
TINYINT	F	T	T	T	T	T
SMALLINT	F	F	T	T	T	T
INT	F	F	F	T	T	T
BIGINT	F	F	F	F	T	T
FLOAT	F	F	F	F	F	T

From/To	BOOLEAN	TINYINT	SMALLINT	INT	BIGINT	FLOAT
DOUBLE	F	F	F	F	F	F
DECIMAL	F	F	F	F	F	F
STRING	F	F	F	F	F	F
VARCHAR	F	F	F	F	F	F
TIMESTAMP	F	F	F	F	F	F
BINARY	F	F	F	F	F	F

Implicit type conversion 2

From/To	DOUBLE	DECIMAL	STRING	VARCHAR	TIMESTAMP	BINARY
BOOLEAN	F	F	F	F	F	F
TINYINT	T	T	T	T	F	F
SMALLINT	T	T	T	T	F	F
INT	T	T	T	T	F	F
BIGINT	T	T	T	T	F	F
FLOAT	T	T	T	T	F	F
DOUBLE	T	T	T	T	F	F
DECIMAL	F	T	T	T	F	F
STRING	T	T	T	T	F	F
VARCHAR	T	T	T	T	F	F
TIMESTAMP	F	F	T	T	T	F
BINARY	F	F	F	F	F	T

T indicates that the type conversion can be performed, while F indicates that the type conversion cannot be performed.

Note

- An unsupported implicit type conversion will cause an exception.
- If the conversion fails, an error is returned.
- Implicit type conversion is automatically performed by MaxCompute based on context. If the types do not match, we recommend that you perform explicit type conversion using cast.
- The rules of implicit type conversion are applied to different specific scopes. In certain scenarios, only part of the rules will take effect. For more information, see the scope of implicit type conversions.

Implicit type conversion with relational operators

Relational operators include equal to (=), not equal to (<>), less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), IS NULL, IS NOT NULL, LIKE, RLIKE, and IN. The implicit conversion rules of LIKE, RLIKE, and IN are different from those of the other relational operators. These three operators are described in a separate section. The rules described in this section do not apply to these three operators. The following table lists implicit conversion rules when different types of data are involved in relational calculations.

Implicit type conversion with relational operators

From/To	BIGINT	DOUBLE	STRING	DATETIME	BOOLEAN	DECIMAL
BIGINT	-	DOUBLE	DOUBLE	N	N	DECIMAL
DOUBLE	DOUBLE	-	DOUBLE	N	N	DECIMAL
STRING	DOUBLE	DOUBLE	-	DATETIME	N	DECIMAL
DATETIME	N	N	DATETIME	-	N	N
BOOLEAN	N	N	N	N	-	N
DECIMAL	DECIMAL	DECIMAL	DECIMAL	N	N	-

Note

- If implicit type conversion is not supported between two values to be compared, the relational operation cannot be completed and an error is returned.
- For more information about relational operators, see Relational operators.

Implicit conversion with special relational operators

Special relational operators are LIKE, RLIKE, and IN.

LIKE and RLIKE are used as follows:

```
source like pattern;
source rlike pattern;
```

Note the following points for the two relational operators in implicit type conversion:

- The source and pattern parameters of LIKE and RLIKE must be of the string type.
- Other types are not supported by this operation and cannot be implicitly converted to the STRING type.
- If the value of source or pattern is NULL, the operation returns NULL.

IN is used as follows:

```
key in (value1, value2,...)
```

The implicit conversion rules of IN are as follows:

- The data types in the value list specified by IN must be consistent.
- If keys and values are compared, the BIGINT, DOUBLE, and STRING types compared are converted to DOUBLE, whereas the DATETIME and STRING types compared are converted to DATETIME. Conversion between other types is not allowed.

Note the following points for the IN operator:

The memory used by the compiler increases with the number of parameters used by the IN operation. An IN operation with 5,000 parameters consumes 17 GB of memory with the GCC compiler. We recommend that you limit the number of parameters to around 1,024. In this case, memory consumption will peak at 1 GB and compilation will only take 39 seconds.

Implicit type conversion with arithmetic operators

Arithmetic operators include plus (+), minus (-), multiplier (*), divider (/), and percent (%). The implicit conversion rules are as follows:

- Only the STRING, BIGINT, DECIMAL, and DOUBLE types can be used in arithmetic operations.
- Before an arithmetic operation, STRING values are implicitly converted to DOUBLE values.
- When an arithmetic operation involves values of both the BIGINT and DOUBLE types, BIGINT values are implicitly converted to DOUBLE values.
- The DATETIME and BOOLEAN types cannot be used in arithmetic operations.

> **Note** For more information about arithmetic operators, see Arithmetic operators.

Implicit conversion with logical operators

Logical operators include AND, OR, and NOT. The implicit conversion rules are as follows:

- Only the BOOLEAN type can be used in logic operations.
- The other types are not supported by logical operations or implicit type conversions.

> **Note** For more information about logical operators, see Logical operators.

1.6.1.4.3. SQL built-in functions

MaxCompute SQL provides a variety of system functions, which can be used to calculate one or more columns of any row and output any type of data.

The implicit conversion rules as follows:

- In a call of a function, if the data type of an input parameter is not consistent with the data type defined in the function, the data type of the input parameter is converted to the function-defined data type.
- The parameters of each built-in SQL function on MaxCompute can have different requirements for implicit type conversion. For more information, see Built-in functions.

1.6.1.4.4. CASE WHEN

This topic describes the implicit conversion rules of CASE WHEN.

The implicit conversion rules of case when are as follows:

- If the returned data types are only bigint and double, they are converted to the double type.
- If data of the string type is also returned, all data types are converted to string. If a data type cannot be converted to string (for example, boolean), an error is returned.
- Conversion between other types is not allowed.

1.6.1.4.5. Partition column

MaxCompute SQL supports partitioned tables. For the definition of partitioned tables, see DDL statements and DML statements. MaxCompute supports partitions of the following types: tinyint, smallint, int, bigint, varchar, and string.

1.6.1.4.6. UNION ALL

The data type, number of column, and column names involved in UNION ALL operation must all be consistent. Otherwise, an error is returned.

1.6.1.4.7. Conversion between string and datetime types

MaxCompute supports conversion between string and datetime types.

The format used in conversion is yyyy-mm-dd hh:mi:ss.ff3.

Value ranges of units

Unit	String (case-insensitive)	Value range
Year	yyyy	0001-9999
Month	mm	01-12
Day	dd	01-28,29,30,31
Hour	hh	00-23
Minute	mi	00-59
Second	ss	00-59
ms	ff3	00-999

> **Note**
>
> - Leading zeros cannot be omitted. For example, 2017-1-9 12:12:12 is an invalid string and cannot be converted into datetime. It must be written as 2017-01-09 12:12:12.
> - Only strings that meet the preceding format requirements can be converted into datetime. For example, cast("2017-12-31 02:34:34" as datetime) converts the "2017-12-31 02:34:34" string into datetime. Similarly, when datetime is converted into strings, the default conversion format is yyyy-mm-dd hh:mi:ss. If you attempt to convert the following examples (or similar strings), the operation will fail and cause an exception.
>
> ```
> cast("2017/12/31 02/34/34" as datetime)
> cast("20171231023434" as datetime)
> cast("2017-12-31 2:34:34" as datetime)
> ```

MaxComppute provides the to_date function, which converts a string type that does not meet the datetime format into datetime type. For more information, see TO_DATE.

1.6.2. Operators

1.6.2.1. Relational operators

This topic describes relational operators in MaxCompute SQL operators.

Relational operators

Operator	Description
A=B	If A or B is NULL, NULL is returned. If A is equal to B, TRUE is returned. Otherwise, FALSE is returned.
A<>B	If A or B is NULL, NULL is returned. If A is not equal to B, TRUE is returned. Otherwise, FALSE is returned.
A<B	If A or B is NULL, NULL is returned. If A is less than B, TRUE is returned. Otherwise, FALSE is returned.
A<=B	If A or B is NULL, NULL is returned. If A is less or equal to B, TRUE is returned. Otherwise, FALSE is returned.
A>B	If A or B is NULL, NULL is returned. If A is greater than B, TRUE is returned. Otherwise, FALSE is returned.
A>=B	If A or B is NULL, NULL is returned. If A is greater than or equal to B, TRUE is returned. Otherwise, FALSE is returned.
A IS NULL	If A is NULL, TRUE is returned. Otherwise, FALSE is returned.
A IS NOT NULL	If A is not NULL, TRUE is returned. Otherwise, FALSE is returned.

Operator	Description
A LIKE B	If A or B is NULL, NULL is returned. A is a string and B is the pattern to be matched. If A matches B, TRUE is returned. Otherwise, FALSE is returned. The percent sign (%) is a wildcard character that matches an arbitrary number of characters. The underscore (_) is a wildcard character that matches a single character. To use these two characters as ordinary characters, use backslashes to escape them: \% and _. 'aaa'like 'a ' = TRUE'aaa' like'a%' = TRUE'aaa'like 'aab' = FALSE'a%b'like 'a\%b' = TRUE'axb'like 'a\%b' = FALSE
A RLIKE B	If A or B is NULL, NULL is returned. A is a string and B is a string constant regular expression. If A matches B, TRUE is returned. Otherwise, FALSE is returned. If B is NULL, the system returns an error and exits.
A IN B	B is a set. If A is NULL, NULL is returned. If A is in B, TRUE is returned. Otherwise, FALSE is returned. If B contains only one element NULL, that is, A IN (NULL), NULL is returned. If B contains NULL, the type of NULL is considered the same as the other elements in B. B must be a constant and have at least one element. All elements must be of the same type.

Double type values have variable precision. We recommend that you do not use the equal sign (=) to compare two double type values. You can subtract between two values of the double type, and then take the absolute value of the result for comparison. When the absolute value is negligible, the two values of the double type are considered equal. For example:

```
abs(0.9999999999 - 1.0000000000) < 0.000000001
-- 0.9999999999 and 1.0000000000 have 10 decimal digits, while 0.000000001 has 9 decimal digits.
-- 0.9999999999 is considered equal to 1.0000000000.
```

Note

- ABS is a built-in function provided by MaxCompute to take the absolute value of its input. For more information, see ABS.
- A value of the double type in MaxCompute can retain 16 valid digits.

1.6.2.2. Arithmetic operators

This topic describes arithmetic operators in MaxCompute SQL operators.

Arithmetic operators

Operator	Description
A + B	If A or B is NULL, NULL is returned. Otherwise, the result of A + B is returned.
A - B	If A or B is NULL, NULL is returned. Otherwise, the result of A - B is returned.
A * B	If A or B is NULL, NULL is returned. Otherwise, the result of A * B is returned.
A / B	If A or B is NULL, NULL is returned. Otherwise, the result of A / B is returned. If both A and B are of the bigint type, the result is of the double type.
A % B	If A or B is NULL, NULL is returned. Otherwise, the result of A % B is returned.
+A	A is returned.
-A	If A is NULL, NULL is returned. Otherwise, -A is returned.

Note

- Only values of the string, bigint, double, and decimal types can be used in arithmetic operations. Values of the datatime and boolean types are not allowed in these operations.
- Before the operation, values of the string type are converted to the double type by implicit type conversion.
- When values of the bigint and double types are involved in an operation, values of the bigint type are converted to the double type by implicit type conversion first. The returned result is a value of the double type.
- When both A and B are of the bigint type, the returned result of A / B is a value of the double type. The returned results of the other arithmetic operations are values of the bigint type.

1.6.2.3. Bitwise operators

This topic describes bitwise operators in MaxCompute SQL operators.

Bitwise operators

Operator	Description
A & B	Returns the bitwise AND result of A and B. For example, 1 & 2 returns 0 and 1 & 3 returns 1. The bitwise AND result of NULL in combination with another value is always NULL. A and B must be of the bigint type.
A | B	Returns the bitwise OR result of A and B. For example, 1 | 2 returns 3 and 1 | 3 returns 3. The bitwise OR result of NULL in combination with another value is always NULL. A and B must be of the bigint type.

Notice Bitwise operators only support bigint type data and do not support implicit type conversion.

1.6.2.4. Logical operators

This topic describes logical operators in MaxCompute SQL operators.

Logical operators

Operator	Description
A and B	TRUE and TRUE = TRUE
	TRUE and FALSE = FALSE
	FALSE and TRUE = FALSE
	FALSE and NULL = FALSE
	FALSE and FALSE = FALSE
	NULL and FALSE = FALSE
	TRUE and NULL = NULL
	NULL and TRUE = NULL
	NULL and NULL = NULL
A or B	TRUE or TRUE = TRUE
	TRUE or FALSE = TRUE
	FALSE or TRUE = TRUE
	FALSE or NULL = NULL
	NULL or FALSE = NULL
	TRUE or NULL = TRUE
	NULL or TRUE = TRUE
	NULL or NULL = NULL
NOT A	If expression A is NULL, NULL is returned.
	If expression A is TRUE, FALSE is returned.
	If expression A is FALSE, TRUE is returned.

> **Note** Only data of the boolean type can be involved in logic operations. These operations do not support implicit type conversion.

1.6.3. DDL statements

1.6.3.1. Table operations

1.6.3.1.1. Create a table (CREATE TABLE)

This topic describes how to execute a DDL statement to create a table.

Syntax

```
create table [if not exists] table_name
[(col_name data_type [DEFAULT value] [comment col_comment], ...)]
[comment table_comment]
[partitioned by (col_name data_type [comment col_comment], ...)]
[STORED AS AliOrc]-- Specify the storage format of the table. You can specify AliORC only for newly cre
ated internal tables.
[lifecycle days]
[as select_statement]
create table [if not exists] table_name like existing_table_name
```

Note

- Table and column names are not case-sensitive.
- If you do not specify the IF NOT EXISTS option and another table with the same name exists, an error is returned. If you specify this option, a message that indicates the operation succeeded is returned. The message is returned regardless of whether an existing table with the same name exists. The message is returned even if the schema of the existing table is different from that of the table you want to create. In addition, the metadata of the existing table does not change.
- A table can contain a maximum of 1,200 column definitions.
- Supported data types are BIGINT, DOUBLE, BOOLEAN, DATETIME, DECIMAL, STRING, ARRAY <T>, and MAP <T1, T2>.

  Note If you need to use the following newly supported data types: TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAMP, or BINARY, you must add the `set odps.sql.type.system.odps2=true;` flag before the CREATE TABLE statement. Then, commit them for execution.

- MaxCompute allows you to specify the default value of a column by using DEFAULT value. If the value of a column is not specified in an INSERT operation, the default value is used for this column.
- A table or column name cannot contain special characters. It can contain only lowercase letters, uppercase letters, digits, or underscores (_). A name must start with a letter and can be up to 128 bytes in length.
- The partitioned by option specifies the partition field. The value can only be a string. The name of a partition key column cannot contain double-byte characters. It must start with a letter, either in lowercase or uppercase, followed by letters or digits. The name can be up to 128 bytes in length. The name can contain the following special characters: ! _ : $.# @ and spaces. Other characters, such as \t, \n, and /, are considered undefined characters. After you use partition fields to define partitions for a table, a full table scan is no longer triggered when you add partitions, update partition data, or read partition data. This improves processing efficiency.
- A comment is a valid string that can be up to 1,024 bytes in length.
- The lifecycle option indicates the lifecycle of the table in days. The CREATE TABLE LIKE statement does not replicate the lifecycle attribute from the source table.
- Theoretically, a source table can have up to six levels of partitions. Use as few partitions as possible to avoid extreme table expansion of storage.
- You can configure the maximum number of table partitions for a project. The default maximum number is 60,000.
- STORED AS specifies the storage format of the table. The default value is CFile2. AliORC in C++ is now available. It is developed by the MaxCompute storage team. AliORC is fully compatible with the open source Optimized Row Columnar (ORC). Compared with CFile2, AliORC frees up more than 10% of extra storage space and improves read performance by more than 20%.

Examples

The following example describes how to create a table named sale_detail to store sales records. The sale_date and region columns of the table are used as partition key columns.

```
create table if not exists sale_detail( shop_name string,
customer_id string,
total_price double)
partitioned by (sale_date string,region string);
-- Create a partitioned table named sale_detail.
```

Use the following `create table...as select...` statement to create a table and replicate data to it:

```
create table sale_detail_ctas1 as select * from sale_detail;
```

> **Note** If the sale_detail table contains data, all the data is replicated to sale_detail_ctas1. The sale_detail table is a partitioned table. However, the table created by the `create table...as select...` statement does not replicate the partition attribute of sale_detail. Partition key columns in sale_detail become standard columns in sale_detail_ctas1. Therefore, sale_detail_ctas1 is a non-partitioned table that has five columns.

In the `create table...as select...` statement, if you use constants as column values in the SELECT clause, we recommend that you specify column aliases:

```
create table sale_detail_ctas2 as select shop_name,
customer_id, total_price,
'2017' as sale_date,
'China' as region from sale_detail;
```

> **Note**
>
> If you do not specify column aliases, the fourth and fifth columns of sale_detail_ctas3 created in the following example are automatically named _c3 and _c4.

```
create table sale_detail_ctas3 as select shop_name,
customer_id, total_price, '2017',
'China'
from sale_detail;
```

In this case, to reference sale_detail_ctas3 again, you must enclose _c3 and _c4 in two pairs of grave accents (`). If you execute the select c3, _c4 from sale_detail_ctas3 statement, an error is returned. The column name in a MaxCompute SQL statement cannot start with underscores (_). Therefore, grave accents (`) must be used. We recommend that you use aliases to avoid this issue.

```
select `_c3`, `_c4` from sale_detail_ctas3;
```

To ensure that the destination table has the same schema as the source table, use the following `create table...like` statement:

```
create table sale_detail_like like sale_detail;
```

> **Note** The schema of sale_detail_like is exactly the same as that of sale_detail. Both tables have the same attributes, such as column names, column comments, and table comments, except for the lifecycle. However, data in sale_detail is not replicated to sale_detail_like.

MaxCompute allows you to execute the DESC statement to view table information.

```
desc <table_name>;
desc extended <table_name>;-- View table information and extended information.
```

MaxCompute allows you to use the SHOW CREATE TABLE statement to generate a DDL statement for table creation. This facilitates the SQL-based rebuild of the table schema.

```
SHOW CREATE TABLE <table_name>;
```

1.6.3.1.2. Delete a table

This topic describes how to run a DDL statement to delete a table.

Command syntax:

```
drop table [if exists] table_name;
```

Note If the command is run without the IF EXISTS option and the table does not exist, an exception is returned. With this option, a success is returned regardless of whether the table exists.

Example:

```
create table sale_detail_drop like sale_detail; drop table sale_detail_drop;
-- If the table exists, a success is returned. If not, an exception is returned.
drop table if exists sale_detail_drop2;
-- A success is returned regardless of whether sale_detail_drop2 exists.
```

1.6.3.1.3. Rename a table

This topic describes how to run a DDL statement to rename a table.

Command syntax:

```
alter table table_name rename to new_table_name;
```

Note

- The rename operation only changes the table name, not the table data.
- If the table specified by new_table_name already exists, an error is returned.
- If the table specified by table_name does not exist, an error is returned.

Example:

```
create table sale_detail_rename1 like sale_detail;
alter table sale_detail_rename1 rename to sale_detail_rename2;
```

1.6.3.1.4. Modify the comment of a table

This topic describes how to run a DDL statement to modify the comment of a table.

Command syntax:

```
alter table table_name set comment 'tbl comment';
```

> Note
>
> - table_name must be an existing table.
> - A comment can contain a maximum of 1,024 bytes.

Example:

```
alter table sale_detail set comment 'new coments for table sale_detail';
```

You can run the desc command to view the modified comment in the table. For more information, see Obtain table information.

1.6.3.1.5. Modify the lifecycle of a table

MaxCompute provides the lifecycle management function to release storage space and simplify the data clearance process. This topic describes how to run a DDL statement to modify the lifecycle of a table.

Command syntax:

```
alter table table_name set lifecycle days;
```

> Note
>
> - The days parameter indicates the lifecycle of a table. Unit: days. It must be a positive integer.
> - If the table specified by table_name is a non-partitioned table, and is not modified in the period specified by the days parameter since the last modification date, MaxCompute automatically clears the table (similar to the DROP TABLE operation). In MaxCompute, the LastDataModifiedTime value of a table is updated each time data in the table is modified. MaxCompute determines whether to clear a table based on its LastDataModifiedTime and lifecycle settings.
> - If the table specified by table_name is a partitioned table, MaxCompute determines whether to clear each partition based on the LastDataModifiedTime value. Unlike non-partitioned tables, a partitioned table is not deleted after the last partition is reclaimed.
> - You can configure a lifecycle for tables, but not for partitions.
> - You can specify a lifecycle when creating a table.

Example:

```
create table test_lifecycle(key string) lifecycle 100;
-- Create a table named test_lifecycle with a lifecycle of 100 days.
alter table test_lifecycle set lifecycle 50;
-- Change the lifecycle of the test_lifecycle table to 50 days.
```

1.6.3.1.6. Disable or restore the lifecycle feature

In some cases, if you do not want some partitions to be automatically reclaimed based on the lifecycle feature, you can disable the lifecycle feature for these partitions. This topic describes how to execute DDL statements to disable or restore the lifecycle feature.

Syntax

```
ALTER TABLE table_name partition[partition_spec] ENABLE|DISABLE LIFECYCLE;
```

> **Note**
>
> - TABLE DISABLE LIFECYCLE
>   - It prevents the reclamation of a table and its partitions based on the lifecycle feature. This option has a higher priority than partition_spec enable lifecycle.
>   - The lifecycle settings and the partition_spec enable/disable flag of a table are retained.
>   - You can still modify the lifecycle settings of a table and its partitions.
> - TABLE ENABLE LIFECYCLE
>   - After the lifecycle feature is enabled again, a table and its partitions can be reclaimed based on the lifecycle feature. By default, the lifecycle settings of the current table and its partitions are used.
>   - Before you restore the lifecycle feature for a table and its partitions, you can configure new lifecycles for the table and its partitions. This prevents data from being mistakenly reclaimed due to the use of the previous settings.

Example

```
ALTER TABLE trans PARTITION(dt='20191111') DISABLE LIFECYCLE;
```

1.6.3.1.7. Modify the LastDataModifiedTime value of a table

MaxCompute SQL supports the TOUCH operation, which allows you to modify the LastDataModifiedTime value of a table. This operation changes the LastDataModifiedTime value of a table to the current time. This topic describes how to run a DDL statement to modify the LastDataModifiedTime value of a table.

Command syntax:

```
alter table table_name touch;
```

> **Note**
> - If the specified table_name does not exist, an error is returned.
> - This operation modifies the LastDataModifiedTime value of the table. In this case, MaxCompute considers a change to the table data, and recalculates the lifecycle.

For more information about how to modify the LastDataModifiedTime value of a partition, see Modify the LastDataModifiedTime value of a partition.

1.6.3.1.8. Clear data from a non-partitioned table

This topic describes how to run a DDL statement to clear data from a non-partitioned table.

Command syntax:

```
TRUNCATE TABLE table_name;
```

> **Note** This statement is used to clear data from a specified non-partitioned table. To clear data from a partitioned table, run the `ALTER TABLE table_name DROP PARTITION (partition_spec)` statement.

1.6.3.1.9. Archive table data

This topic describes how to run a DDL statement to archive the data of a table.

If a project does not have enough space, you can use the table archiving feature in MaxCompute to compress data by about 50%. The archiving feature uses a compression algorithm with a higher compression ratio. It saves data as redundant array of independent disks (RAID) files. Data is no longer simply stored in three copies. Instead, six copies and three check blocks are maintained to increase the effective storage ratio from 1:3 to 1:1.5. The archive feature consumes only half of the usual physical space.

However, this feature comes at a price. If a data block or machine is damaged, the time required to restore the data is longer, and the read performance is affected. Therefore, this feature is suitable for compressing cold data for storage. For example, you can store large volumes outdated log data as RAID files for a long time.

Command syntax:

```
ALTER TABLE [table_name] <PARTITION(partition_name='partition_value')> ARCHIVE;
```

Example:

```
alter table my_log partition(ds='20170101') archive;
```

Command output:

```
Summary:
table name: test0128 /pt=a instance count: 1 run time: 21
before merge, file count: 1 file size: 456 file physical size: 1368
after merge, file count: 1 file size: 512 file physical size: 768
```

> **Note**
>
> The output shows the changes in logical size and physical size during the archiving process. In the archiving process, multiple small files are automatically merged. After the archive operation is complete, you can run the `desc extended` command to check whether the data in the partition has been archived, and view the physical space usage:

```
desc extended my_log partition(ds='20170101');
+------------------------------------------------------------------------------------+
PartitionSize: 512 |
+------------------------------------------------------------------------------------+
CreateTime: 2017-01-28 07:05:20 |
LastDDLTime: 2017-01-28 07:05:20 |
LastModifiedTime: 2017-01-28 07:05:21 |
+------------------------------------------------------------------------------------+
```

1.6.3.1.10. Forcibly delete data from a table (partition)

If you need to forcibly and irrecoverably delete data from a table or partition to immediately release storage space, you can perform the deletion operation with the PURGE option. This topic describes how to run a DDL statement to forcibly delete data from a table (partition).

Command syntax:

```
DROP TABLE tblname PURGE;
ALTER TABLE tblname DROP PARTITION(part_spec) PURGE;
```

Example:

```
drop table my_log purge;
alter table my_log drop partition (ds='20170618') purge;
```

1.6.3.2. View-based operation

1.6.3.2.1. Create a view

This topic describes how to run a DDL statement to create a view.

Command syntax:

```
create [or replace] view [if not exists] view_name
[(col_name [comment col_comment], ...)]
[comment view_comment]
[as select_statement]
```

> **Note**
>
> - To create a view, you must have read permissions on the table referenced by the view. Views in MaxCompute are not materialized views. View operations involve accessing data of referenced tables. Note that changes to your permission on the referenced table can result in changes to your permission on the view.
> - A view can contain only one valid SELECT statement.
> - A view can reference other views but cannot reference itself. Circular reference is not supported.
> - You cannot write data to a view. For example, the INSERT INTO and INSERT OVERWRITE operations do not work on views.
> - If the table referenced by a view changes, you may no longer be able to access the view. For example, a view becomes inaccessible after the table it references is deleted. You must maintain the mappings between referenced tables and views properly.
> - If the CREATE VIEW statement is run without the IF NOT EXISTS option and the view already exists, an exception is returned. In this case, you can run the CREATE VIEW or REPLACE VIEW statement to recreate a view. The permissions on the recreated view remain unchanged.

Example:

```
create view if not exists sale_detail_view
(store_name, customer_id, price, sale_date, region)
comment 'a view for table sale_detail'
as select * from sale_detail;
```

1.6.3.2.2. Delete a view

This topic describes how to run a DDL statement to delete a view.

Command syntax:

```
drop view [if exists] view_name;
```

> **Note** If the command is run without the IF EXISTS option and the view does not exist, an error is returned.

Example:

```
drop view if exists sale_detail_view;
```

1.6.3.2.3. Rename a view

This topic describes how to run a DDL statement to rename a view.

Command syntax:

```
alter view view_name rename to new_view_name;
```

> **Note** If a view with the same name already exists, an error is returned.

Example:

```
create view if not exists sale_detail_view
(store_name, customer_id, price, sale_date, region)
comment 'a view for table sale_detail'
as select * from sale_detail;
alter view sale_detail_view rename to market;
```

1.6.3.3. Column and partition operations

1.6.3.3.1. Add a partition (ADD PARTITION)

This topic describes how to use a DDL statement to add a partition.

Syntax

```
alter table table_name add [if not exists] partition partition_spec;-- Add a partition.
alter table table_name add [if not exists] partition partition_spec [PARTITION partition_spec PARTITION
partition_spec...];-- Add multiple partitions at a time.
partition_spec:(partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...)
```

> **Note**
>
> - If you do not specify the IF NOT EXISTS option and another partition with the same name exists, an error is returned.
> - A MaxCompute table can contain a maximum of 60,000 partitions.
> - To add a partition to a table that has multi-level partitions, you must specify all partitioning column values.

Examples

The following examples show how to add partitions to the sale_detail table:

```
alter table sale_detail add if not exists partition (sale_date='201712', region='hangzhou');
-- Add a partition to store the sales records of the China (Hangzhou) region for December 2017.
alter table sale_detail add if not exists partition (sale_date='201712', region='shanghai');
-- Add a partition to store the sales records of the China (Shanghai) region for December 2017.
alter table sale_detail add if not exists partition(sale_date='20171011');
-- Specify only the sale_date partition. An error is returned.
alter table sale_detail add if not exists artition(region='shanghai');
-- Specify only the region partition. An error is returned.
```

1.6.3.3.2. Delete a partition (DROP PARTITION)

This topic describes how to use a DDL statement to delete a partition.

Syntax

```
alter table table_name drop [if exists] PARTITION partition_spec; -- Delete a partition.
alter table table_name drop [if exists] PARTITION partition_spec,PARTITION partition_spec,[PARTITION
partition_spec....] ;-- Delete multiple partitions at a time.
partition_spec:: (partition_col1 = partition_col_value1, partition_col2 = partiton_col_value2, ...)
```

> **Note** If you do not specify the IF EXISTS option and the partition you want to delete does not exist, an error is returned.

Example

Execute the following statement to delete a partition from the sale_detail table:

```
alter table sale_detail drop partition(sale_date='201712',region='hangzhou');
-- The sales records of the China (Hangzhou) region for December 2017 are deleted.
```

1.6.3.3.3. Add a column

This topic describes how to add a column by using a DDL statement.

Command syntax:

```
alter table table_name add columns (col_name1 type1, col_name2 type2...)
```

Note

- A column can only be one of the following types: bigint, double, boolean, datetime, decimal, string, tinyint, smallint, int, float, varchar, binary, timestamp, array, map, or struct.
- You can create up to 1,200 columns in a single table in MaxCompute.

1.6.3.3.4. Change a column name

This topic describes how to run a DDL statement to change a column name.

Command syntax:

```
alter table table_name change column old_col_name rename to new_col_name;
```

Note

- You must specify an existing column for old_col_name.
- You cannot name a column in the table new_col_name.

1.6.3.3.5. Modify the comment of a column or partition

This topic describes how to run a DDL statement to modify the comment of a column or partition.

Command syntax:

```
alter table table_name change column col_name comment 'comment';
```

Note

- The comment cannot exceed 1,024 bytes.
- The data type and position of a column cannot be changed.

1.6.3.3.6. Modify the LastDataModifiedTime value of a partition

MaxCompute SQL supports the TOUCH operation, which allows you to modify the LastDataModifiedTime value of a partition. This operation changes the LastDataModifiedTime value of a partition to the current time. This topic describes how to run a DDL statement to modify the LastDataModifiedTime value of a partition.

Command syntax:

```
alter table table_name touch partition(partition_col='partition_col_value', ...);
```

> Note
>
> - If the specified table_name or partition_col does not exist, an error is returned.
> - If the specified partition_col_value does not exist, an error is returned.
> - This operation modifies the LastDataModifiedTime value of the table. In this case, MaxCompute considers a change to the table or partition value, and recalculates the lifecycle.

For more information about how to modify the LastDataModifiedTime value of a table, see Modify the LastDataModifiedTime value of a table.

1.6.3.3.7. Modify partition values

MaxCompute SQL provides the RENAME operation, which allows you to modify partition values of a table. This topic describes how to run a DDL statement to modify partition values.

Command syntax:

```
ALTER TABLE table_name PARTITION (partition_col1 = partition_col_value1, partition_col2 = partiton_col
_value2, . ..)
RENAME TO PARTITION (partition_col1 = partition_col_newvalue1, partition_col2 = partiton_col_newvalu
e2, ...);
```

> Note
>
> - This command cannot modify the names of partition columns. It can only modify the values of the columns.
> - To modify the values in one or more partitions in the case of multi-level partitions, you must specify values of partitions at each level.

1.6.3.3.8. Merge partitions

MaxCompute allows you to merge multiple partitions in a table into one partition and delete original partitions.

Syntax

```
ALTER TABLE <tableName> MERGE [IF EXISTS] PARTITION(<predicate>) [, PARTITION(<predicate2>) ...] O
VERWRITE PARTITION(<fullPartitionSpec>) [PURGE];
```

> Note
>
> - If you do not specify the IF EXISTS option and the partition you want to merge does not exist, an error is returned.
> - If you specify the IF EXISTS option but no partitions meet the merge conditions, no new partitions are generated.
> - If source data is concurrently modified by operations such as INSERT, RENAME, or DROP when you execute the preceding statement, an error is returned even though you have specified the IF EXISTS option.
> - If the PURGE attribute is specified, merged partitions cannot be restored by using the Kunlunjing.

Limits and troubleshooting

- External tables, shard tables, and tables with extreme storage are not supported. Xlib or Algo tables that depend on the file order are not supported. If you merge partitions of a clustered table, the clustered attribute is removed from the partitions.
- Hash operations are performed by CatalogServer on tables to merge partitions. A capacity limit is imposed on merged partitions. A hard link in the Apsara Distributed File System can have a maximum of seven replicas.
- You can merge a maximum of 4,000 partitions at a time.
- The number of partitions that can wait on CatalogServer to be merged is 10 million.
- If an error that indicates CatalogServer is busy occurs, try again later.
- If a hard link in the Apsara Distributed File System is faulty, purge the recycle bin and then try again.

Example

The following code shows the partitions and data of the tb_test table:

```
odps@ jet_zwz>list partitions tb_test;
ds=20181101/hh=00/mm=00
ds=20181101/hh=00/mm=10
ds=20181101/hh=10/mm=00
ds=20181101/hh=10/mm=10
OK
odps@ jet_zwz>read intpstringstringstring;
+------------+------------+------------+------------+
| value    | ds       | hh       | mm       |
+------------+------------+------------+------------+
| 1        | 20181101   | 00       | 00       |
| 1        | 20181101   | 00       | 10       |
| 1        | 20181101   | 10       | 00       |
| 1        | 20181101   | 10       | 10       |
+------------+------------+------------+------------+
```

Execute the following statement to merge all partitions that meet the hh='00' condition into the ds=20181101/hh=00/mm=00 partition:

```
odps@ jet_zwz>alter table intpstringstringstring merge partition(hh='00') overwrite partition(ds='2018
1101', hh='00', mm='00');
ID = 20190404025755844g80qwa7a
OK
```

Execute the following statement to view the partitions of the table after they are merged:

```
odps@ jet_zwz>list partitions intpstringstringstring;
ds=20181101/hh=00/mm=00
ds=20181101/hh=10/mm=00
ds=20181101/hh=10/mm=10
OK
```

Data in two partitions that meet the hh='00' condition is merged into the ds=20181101/hh=00/mm=00 partition.

```
odps@ jet_zwz>read intpstringstringstring;
+------------+------------+------------+------------+
| value     | ds         | hh         | mm         |
+------------+------------+------------+------------+
| 1         | 20181101   | 00         | 00         |
| 1         | 20181101   | 00         | 00         |
| 1         | 20181101   | 10         | 00         |
| 1         | 20181101   | 10         | 10         |
+------------+------------+------------+------------+
```

When you merge partitions, you can specify multiple predicate conditions. For example, you can execute the following statement to merge all the partitions that remain to the ds=20181101/hh=00/mm=00 partition:

```
odps@ jet_zwz>alter table intpstringstringstring merge if exists partition(ds='20181101', hh='00', mm='
00'), partition(ds='20181101', hh='10', mm='00'),  partition(ds='20181101', hh='10', mm='10') overwrite part
ition(ds='20181101', hh='00', mm='00') purge;
ID = 20190404034632854g431sqzt2
OK
odps@ jet_zwz>show partitions intpstringstringstring;
ds=20181101/hh=00/mm=00
OK
```

1.6.4. DML statements

1.6.4.1. INSERT statement

1.6.4.1.1. Update the data of a table

This topic describes how to run an INSERT statement to update the data of a table.

The INSERT OVERWRITE and INSERT INTO statements are commonly used for data processing in MaxCompute SQL. They are used to save the computing results in the target table for the next computing. The INSERT INTO statement adds data to a table or partition. The INSERT OVERWRITE statement clears the original data before inserting data to a table or partition.

Command syntax:

```
insert overwrite|into table tablename [partition (partcol1=val1, partcol2=val2 ...)] select_statement
from from_statement;
```

> **Note** The INSERT syntax in MaxCompute is different from that in MySQL or Oracle. In MaxCompute, INSERT OVERWRITE or INSERT INTO must be followed by the keyword TABLE, not directly by the table name.

Example:

The following example calculates the sales of different regions in the sale_detail table.

```
create table sale_detail_insert like sale_detail;
alter table sale_detail_insert add partition(sale_date='2017', region='china');
insert overwrite table sale_detail_insert partition (sale_date='2017', region='china') select shop_name,
customer_id, total_price from sale_detail;
```

> **Note** When data is updated using an INSERT operation, the mapping between the source and target tables depends on the column sequence in the SELECT clause, instead of the mapping of column names between both tables.

The following statement is also valid:

```
insert overwrite table sale_detail_insert partition (sale_date='2017', region='china')
select customer_id, shop_name, total_price from sale_detail;
-- When the sale_detail_insert table is created, the column sequence is shop_name string, customer_id
string, and total_price bigint.
-- When data in sale_detail is inserted to sale_detail_insert, the insertion sequence is customer_id, sh
op_name, and total_price.
-- In this case, data in sale_detail.customer_id is inserted into sale_detail_insert.shop_name.
-- Data in sale_detail.shop_name is inserted into sale_detail_insert.customer_id.
```

When data is inserted into a partitioned table, the partition columns cannot appear in the SELECT list.

```
insert overwrite table sale_detail_insert partition (sale_date='2017', region='china') select shop_name,
customer_id, total_price, sale_date, region from sale_detail;
-- An error is returned, because partition columns (sale_date and region) cannot appear in an INSERT s
tatement for a static partition.
```

1.6.4.1.2. Output data to multiple objects

This topic describes how to run the INSERT statement to output data to multiple objects.

MaxCompute SQL allows you to insert data to different result tables or partitions by using one SQL statement.

Command syntax:

```
from from_statement
insert overwrite | into table tablename1 [partition (partcol1=val1, partcol2=val2 ...)] select_statement1
[insert overwrite | into table tablename2 [partition ...] select_statement2]
```

> **Note**
>
> - A SQL statement typically supports up to 256 outputs. A syntax error is returned if more than 256 outputs are specified.
> - In a MULTI INSERT statement, you can specify a target partition in a partitioned table or specify a non-partitioned table only once.
> - The INSERT OVERWRITE and INSERT INTO operations cannot be performed simultaneously on different partitions in a partitioned table. Otherwise, an error is returned.

Example:

```
create table sale_detail_multi like sale_detail;
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2016', region='china' ) select shop_name,
customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2017', region='china' ) select shop_name,
customer_id, total_price;
-- A success is returned. Data of the sale_detail table is inserted into the sale records of the China reg
ion in 2016 and 2017 in the sales table.
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2017', region='china' ) select shop_name,
customer_id, total_price
insert overwrite table sale_detail_multi partition (sale_date='2017', region='china' ) select shop_name,
customer_id, total_price;
-- An error is returned. The same partition appears more than once.
from sale_detail
insert overwrite table sale_detail_multi partition (sale_date='2016', region='china' )
select shop_name, customer_id, total_price
insert into table sale_detail_multi partition (sale_date='2017', region='china' ) select shop_name, custo
mer_id, total_price;
-- An error is returned. The INSERT OVERWRITE and INSERT INTO operations cannot be performed simul
taneously on different partitions in a partitioned table.
```

1.6.4.1.3. Output data to a dynamic partition

This topic describes how to use the INSERT statement to output data to a dynamic partition.

When you run the INSERT OVERWRITE statement on a partitioned table, you can specify the partition values in the statement. Another flexible method is to specify partition column names instead of setting partition values. In the meantime, specify the partition values in the corresponding columns of a SELECT clause.

Command syntax:

```
insert overwrite table tablename partition (partcol1, partcol2 ...) select_statement from from_stateme
nt;
```

Note

- When you run a SQL dynamic partition statement in a distributed environment, a single process can output up to 512 dynamic partitions. If the number of dynamic partitions exceeds this limit, an exception is returned.
- Currently, a SQL dynamic partition statement can generate up to 2,000 dynamic partitions. If the number of dynamic partitions exceeds this limit, an exception is returned.
- The dynamic partition values cannot be NULL. Otherwise, an exception is returned.
- If a target table has multi-level partitions, you can specify some partitions as static partitions in an INSERT statement. However, the static partitions must be high-level partitions.

Example:

```
create table total_revenues (revenue bigint) partitioned by (region string); insert overwrite table tota
l_revenues partition(region)
select total_price as revenue, region from sale_detail;
```

Note In the preceding example, you do not know which partitions are generated before running the SQL statement. The partitions generated are determined by the value of the region field after the execution of the SELECT statement. This is why the partitions are called dynamic partitions.

Other examples:

```
create table sale_detail_dypart like sale_detail;
insert overwrite table sale_detail_dypart partition (sale_date, region) select * from sale_detail;
-- A success is returned.
insert overwrite table sale_detail_dypart partition (sale_date='2017', region) select shop_name,custom
er_id,total_price,region from sale_detail;
-- A success is returned. The table has multi-level partitions. Specify a primary partition.
insert overwrite table sale_detail_dypart partition (sale_date='2017', region) select shop_name,custom
er_id,total_price from sale_detail;
-- An error is returned. The inserted dynamic partition must be in the SELECT list.
insert overwrite table sales partition (region='china', sale_date) select shop_name,customer_id,total_p
rice,region from sale_detail;
-- An error is returned. You cannot specify only low-level partitions when dynamically inserting high-le
vel partitions.
```

1.6.4.2. SELECT statement

1.6.4.2.1. SELECT

This topic describes how to use the SELECT statement.

Syntax

```
select [all | distinct] select_expr, select_expr, ... from table_reference
[where where_condition] [group by col_list]
[order by order_condition]
[distribute by distribute_condition [sort by sort_condition] ] [limit number]
```

Take note of the following points when you execute the SELECT statement:

- The SELECT statement reads data from a table. You can specify the names of the columns you want to read or use an asterisk (*) to represent all columns.

  Examples

  ```
  select * from sale_detail;
  -- Read data from all columns in the sale_detail table.
  select shop_name from sale_detail;
  -- Read data from the shop_name column in the sale_detail table.
  ```

  > **Note** The SELECT statement can only return a maximum of 1,000 rows of results. However, no such limits are imposed when SELECT is used as a clause. If SELECT is used as a clause, the clause returns all results in response to the query from the upper layer. To obtain more than 1,000 rows of results by using the SELECT statement, you must use Tunnel to download the entire table or a temporary table returned by the SELECT operation. For more information, see MaxCompute Tunnel.

- You can use the WHERE clause to specify filter conditions.

  Examples

  ```
  select * from sale_detail where shop_name like 'hang%';
  ```

  The following table describes filter conditions supported by the WHERE clause.

Filter conditions

Filter condition	Description
>, < , =, >=, <=, <>	/
like, rlike	/
in, not in	If a subquery is added after the IN or NOT IN condition, only one column is returned for the subquery and the number of return values cannot exceed 1,000.

You can specify partitions in the WHERE clause of the SELECT statement to avoid a full table scan.

Examples

```
select sale_detail.* from sale_detail
where sale_detail.sale_date >= '2015' and sale_detail.sale_date <= '2017';
```

> **Notice** To check whether partition pruning takes effect, execute the `EXPLAIN SELECT` statement. A common user-defined function (UDF) or the method that is used to specify partition conditions in a JOIN operation can cause partition pruning to fail to take effect.

UDFs support partition pruning. These UDFs are executed as small jobs and then replaced with the execution results.

You can use one of the following methods:

- Add an annotation to the UDF class when you write a UDF.

  ```
  @com.aliyun.odps.udf.annotation.UdfProperty(isDeterministic=true)
  ```

  > **Notice** com.aliyun.odps.udf.annotation.UdfProperty defines that the version of referenced odps-sdk-udf must be 0.30.x or later in odps-sdk-udf.jar.

- Add the `set odps.sql.udf.ppr.deterministic = true;` flag before SQL statements. Then, all UDFs in the SQL statements are considered deterministic.

  > **Note** This method is used with limits. This method backfills partitions with execution results. A maximum of 1,000 partitions can be backfilled. If an annotation is added to the UDF class, an error that indicates more than 1,000 partitions are backfilled may be returned. If you want to ignore the error, add the `set odps.sql.udf.ppr.to.subquery = false;` flag to disable this feature globally. After this feature is disabled, UDF-based partition pruning becomes invalid.

The WHERE clause in an SQL statement can include the BETWEEN...AND condition. Example:

```
SELECT sale_detail. * FROM sale_detail
WHERE sale_detail.sale_date between '2017' and '2019';
```

> **Note** The number of conditions that can be specified in the WHERE clause cannot exceed 256.

- Nested subqueries are supported in table_reference.

  Examples

  ```
  select * from (select region from sale_detail) t where region = 'shanghai';
  ```

- DISTINCT: If duplicate rows exist, add DISTINCT before the field to remove duplicate values. In this case, only one value is returned. If you use ALL, all duplicate values are returned. If you do not specify the DISTINCT option, the statement returns all duplicate values, same as the result obtained by using the ALL option.

  Examples

  ```
  select distinct region from sale_detail;
  select distinct region, sale_date from sale_detail;
  -- The DISTINCT option applies to multiple columns. The option takes effect on all columns of the SEL
  ECT statement, instead of a single column.
  ```

- GROUP BY: This clause is used to perform group-based queries. In most cases, this clause is used with aggregate functions. If a SELECT statement includes aggregate functions, the key of the GROUP BY clause can be the names of columns in the input table or an expression composed of input table columns. The key cannot be the aliases of the columns in the output table of the SELECT operation.

  > **Note** If the `set hive.groupby.position.alias=true;` flag is added before SQL statements, integer constants in the GROUP BY clause are considered column numbers in a SELECT operation. Example:
  >
  > ```
  > -- The columns in the sale_detail table are in the format of key-value pairs.
  > select region, sum(total_price) from sale_detail group by 1;
  > -- Equivalent to the following statement:
  > select region, sum(total_price) from sale_detail group by region;
  > ```

  Examples

```
select region from sale_detail group by region;
-- The statement is successfully executed because the name of a column in the input table is used a
s the column in the GROUP BY clause.
select sum(total_price) from sale_detail group by region;
-- The statement is successfully executed because the values in the region column are used to grou
p the table and to return the total sales of each group.
select region, sum(total_price) from sale_detail group by region;
-- The statement is successfully executed because the values in the region column are used to grou
p the table and to return the unique region value and the total sales of each group.
select region as r from sale_detail group by r;
-- An error is returned because the alias of the column in the SELECT operation is used as the colum
n in the GROUP BY clause.
select 'China-' + region as r from sale_detail group by 'China-' + region;
-- A complete expression of the column is required.
select region, total_price from sale_detail group by region;
-- An error is returned because all the columns that do not include aggregate functions in the SELEC
T operation must exist in the GROUP BY clause.
select region, total_price from sale_detail group by region, total_price;
-- The statement can be successfully executed.
```

Note The GROUP BY operation is performed before the SELECT operation during the parsing of SQL statements. Therefore, GROUP BY uses only the column names or expressions of the input table as keys. For more information about aggregate functions, see Aggregate functions.

- ORDER BY: This clause is used for global sorting based on specific columns. To sort records in descending order, use the DESC keyword. The ORDER BY clause must be used with the LIMIT clause because records are globally sorted. In an ORDER BY operation, NULL is considered the lowest of all values. This rule is consistent with MySQL, but is different from Oracle. Different from the GROUP BY clause, the columns in the ORDER BY clause must be the aliases of the columns in the SELECT operation. If you want to query a column but the column alias is not specified in the SELECT operation, the column name is used as the column alias.

  Note If the `set hive.orderby.position.alias=true;` flag is added before SQL statements, integer constants in the ORDER BY clause are considered column numbers in a SELECT operation. Example:

  ```
  -- The columns in the sale_detail table are in the format of key-value pairs.
  select region, sum(total_price) from sale_detail order by 2 limit 100;
  -- Equivalent to the following statement:
  select region, sum(total_price) from sale_detail order by sum(total_price) limit 100;
  ```

  Examples

```
select * from sale_detail order by region;
-- An error is returned because the ORDER BY clause is not used with the LIMIT clause.
select * from sale_detail order by region limit 100;
select region as r from sale_detail order by region;
-- An error is returned because the ORDER BY clause is not followed by a column alias.
select region as r from sale_detail order by r;
```

> **Note** The number in the LIMIT clause is a constant that limits the number of output rows. If a SELECT statement is executed without the LIMIT clause, it can return a maximum of 5,000 rows. The screen display limit may vary with projects and can be configured in the console.

The OFFSET clause can be used with the ORDER BY LIMIT clause to skip the number of rows specified by OFFSET.

Examples

```
SELECT * FROM src ORDER BY key LIMIT 20 OFFSET 10;
-- Sort the rows of the src table in ascending order by key, and return the 11th to 30th rows. OFFSET 10 indicates that the first 10 rows are skipped, and LIMIT 20 indicates that a maximum of 20 rows can be returned.
```

- DISTRIBUTE BY: This clause is used to shard data based on hash values of specific columns. The DISTRIBUTE BY clause must be followed by the alias of an output column from the SELECT operation.

  Examples

  ```
  select region from sale_detail distribute by region;
  -- The statement is successfully executed because the column name is used as the column alias.
  select region as r from sale_detail distribute by region;
  -- An error is returned because the DISTRIBUTE BY clause is not followed by a column alias.
  select region as r from sale_detail distribute by r;
  ```

- SORT BY: This clause is used for partial sorting. The DISTRIBUTE BY clause must be placed before the SORT BY clause. In practice, the SORT BY clause is used to partially sort the results of the DISTRIBUTE BY clause. The SORT BY clause must be followed by the alias of an output column from the SELECT operation.

  Examples

  ```
  select region from sale_detail distribute by region sort by region; select region as r from sale_detail sort by region;
  -- An error is returned because the SORT BY clause does not follow a DISTRIBUTE BY clause.
  ```

- The ORDER BY and GROUP BY clauses cannot be used with the DISTRIBUTE BY and SORT BY clauses. The ORDER BY and GROUP BY clauses must be followed by the alias of an output column from the SELECT operation.

> **Note**
>
> - The key of the ORDER BY, SORT BY, or DISTRIBUTE BY clause must be the alias of an output column from the SELECT operation.
> - The SELECT operation is performed before the ORDER BY, SORT BY, and DISTRIBUTE BY clauses during the parsing of SQL statements. Therefore, only the aliases of output columns from the SELECT operation can be used as keys.

1.6.4.2.2. Subquery

This topic describes how to use the SELECT statement for subquery operations.

A common SELECT statement reads data from multiple tables, for example, select column_1, column_2 ... from table_name. The query object can be another SELECT operation, which is a subquery.

Command syntax:

```
select * from (select shop_name from sale_detail) a;
```

> **Notice** A subquery must have an alias.

Example:

```
create table shop as select * from sale_detail;
select a.shop_name, a.customer_id, a.total_price from
(select * from shop) a join sale_detail on a.shop_name = sale_detail.shop_name;
```

> **Note** In a FROM clause, a subquery can be used as a table, which supports a JOIN operation with other tables or subqueries.

1.6.4.3. UNION statements

1.6.4.3.1. UNION ALL

This topic describes how to execute the SELECT statement to perform the UNION ALL operation.

Syntax

```
select_statement union all select_statement
```

> **Note** The UNION ALL clause is used to combine two or more datasets returned from a SELECT operation into one dataset. If duplicate rows exist in the results, all rows that meet the condition are returned, with duplicate rows retained.

MaxCompute SQL does not support the combination of two top-level query results. To combine them, rewrite them into a subquery.

Format example before rewriting

```
select * from sale_detail where region = 'hangzhou'
union all
select * from sale_detail where region = 'shanghai';
```

Format example after rewriting

```
select * from (
select * from sale_detail where region = 'hangzhou' union all
select * from sale_detail where region = 'shanghai') t;
```

The syntax that uses a pair of parentheses to specify the priority of UNION ALL is supported.

Example:

```
SELECT * FROM src UNION ALL (SELECT * FROM src2 UNION ALL SELECT * FROM src3);
-- Execute the UNION ALL clause for the src2 and src3 tables. Then, execute the UNION ALL clause for t
he src table based on the obtained result.
```

> **Notice**
>
> - For a UNION ALL operation, all subqueries must have the same number of columns, column names, and column types. If the column names are inconsistent, use column aliases.
> - In most cases, MaxCompute allows a UNION ALL operation for a maximum of 256 subqueries. If the limit is exceeded, a syntax error is returned.

1.6.4.4. JOIN statement

1.6.4.4.1. JOIN

This topic describes how to use a JOIN statement.

MaxCompute supports multiple JOIN operations in an SQL statement. JOIN does not support Cartesian products (JOIN without an ON clause).

Syntax

```
join_table:
table_reference join table_factor [join_condition]
| table_reference {left outer|right outer|full outer|inner} join table_reference join_condition
table_reference: table_factor
join_table
table_factor: tbl_name [alias]
table_subquery alias
( table_references )
join_condition:
on equality_expression ( and equality_expression )*
```

> **Note** equality_expression indicates an equality expression.

Take note of the following points when you perform a JOIN operation:

- LEFT OUTER JOIN: returns all rows in the left table, such as shop in the following example. The returned rows include the rows that do not match any rows in the right table, such as sale_detail in the following example.

  Example

  ```
  select a.shop_name as ashop, b.shop_name as bshop from shop a left outer join sale_detail b on a.s
  hop_name=b.shop_name;
  -- Both the shop and sale_detail tables have the shop_name column. You must use aliases to disting
  uish the columns in the SELECT operation.
  ```

- RIGHT OUTER JOIN: returns all rows in the right table, such as sale_detail in the following example. The returned rows include the rows that do not match any rows in the left table, such as shop in the following example.

  Example

  ```
  select a.shop_name as ashop, b.shop_name as bshop from shop a right outer join sale_detail b on a.
  shop_name=b.shop_name;
  -- Both the shop and sale_detail tables have the shop_name column. You must use aliases to disting
  uish the columns in the SELECT operation.
  ```

- FULL OUTER JOIN: returns all rows in both the left and right tables.

  Example

  ```
  select a.shop_name as ashop, b.shop_name as bshop from shop a full outer join sale_detail b on a.s
  hop_name=b.shop_name;
  ```

- INNER JOIN: only returns the rows in which two tables can be mapped. The INNER keyword can be omitted.

  Example

```
select a.shop_name from shop a inner join sale_detail b on a.shop_name=b.shop_name; select a.sho
p_name from shop a join sale_detail b on a.shop_name=b.shop_name;
```

- Join condition: You must use equi-joins and combine conditions by using AND. A maximum of 128 JOIN operations are supported in an SQL statement. You can use non-equi joins or combine conditions by using OR in a MAPJOIN operation.

  Example

```
select a.* from shop a full outer join sale_detail b on a.shop_name=b.shop_name full outer join sale_
detail c on a.shop_name=c.shop_name;
-- A maximum of 128 JOIN operations are supported in an SQL statement.
select a.* from shop a join sale_detail b on a.shop_name <> b.shop_name;
-- An error is returned because MaxCompute does not support non-equi joins.
```

- NATURAL JOIN: In a NATURAL JOIN operation, the conditions used to join two tables are automatically determined based on the common fields in the two tables. MaxCompute supports OUTER NATURAL JOIN. You can use the USING clause so that the JOIN operation returns common fields only once.

  Example

```
-- To join the src table that contains the key1, key2, a1, and a2 columns and the src2 table that cont
ains the key1, key2, b1, and b2 columns, execute the following statement:
SELECT * FROM src NATURAL JOIN src2;
-- Both the src and src2 tables include the key1 and key2 fields. In this case, the preceding statemen
t is equivalent to the following statement:
SELECT src.key1 as key1, src.key2 as key2, src.a1, src.a2, src2.b1, src2.b2 FROM src INNER JOIN src2 ON
src.key1 = src2.key1 AND src.key2 = src2.key2;
```

The syntax that uses a pair of parentheses to specify the priorities of JOIN operations is supported.

Example

```
SELECT * FROM src JOIN (src2 JOIN src3 on xxx) ON yyy;
-- The src2 JOIN src3 operation is executed first. Then, the JOIN operation is performed on the src table
based on the result.
```

1.6.4.4.2. MAPJOIN HINT

This topic describes how to use a MAPJOIN statement to join a large table with one or more small tables.

A MAPJOIN operation is faster than common JOIN operations.

When the volume of data is small, MAPJOIN accelerates the execution process by using SQL to load all the specified small tables into the program memory through the JOIN operation.

Example

```
select /* + mapjoin(a) */ a.shop_name, b.customer_id, b.total_price
from shop a join sale_detail b
on a.shop_name = b.shop_name;
```

> **Notice**
>
> Note the following points when you use a MAPJOIN statement:
>
> - The left table of a LEFT OUTER JOIN clause must be a large table.
> - The right table of a RIGHT OUTER JOIN clause must be a large table.
> - Both the left and right tables of an INNER JOIN clause can be large tables.
> - MAPJOIN cannot be used in a FULL OUTER JOIN clause.
> - MAPJOIN supports small tables in subqueries.
> - If you need to reference a small table or a subquery when using MAPJOIN, you must reference the alias of the table or subquery.
> - In MAPJOIN, you can use non-equi joins or combine multiple conditions by using OR.
> - If MAPJOIN is used, the total memory occupied by all the small tables cannot exceed 512 MB. However, you can use the *odps.sql.mapjoin.memory.max* parameter to raise this limit up to 2,048 MB.
>
>   The limit here refers to the original size of data. If you run the desc command to obtain the compressed size, you must multiply it by the compression ratio.

In MaxCompute SQL, you cannot use non-equi joins or the OR logic in the ON condition. However, you can do this in MAPJOIN. Example:

```
select /*+ mapjoin(a) */ a.total_price, b.total_price
from shop a join sale_detail b
on a.total_price < b.total_price or a.total_price + b.total_price < 500;
```

1.6.4.5. EXPLAIN statement

This topic describes the EXPLAIN statement in DML statements of MaxCompute SQL.

MaxCompute SQL provides the EXPLAIN operation, which displays the description of the ultimate execution plan structure of DML statements. An execution plan is the program that is ultimately used to execute SQL semantics.

Command syntax:

```
EXPLAIN <DMLquery>;
```

> **Note**
>
> The execution result of an EXPLAIN statement includes the following:
>
> - Dependencies between all the jobs of this DML statement.
> - Dependencies between all the tasks of each job.
> - All operator dependency structures in a task.

Example:

```
EXPLAIN
SELECT abs(a.key), b.value FROM src a JOIN src1 b ON a.value = b.value;
```

The EXPLAIN statement output includes the following:

- **The first part is the dependency between jobs.**

  Command output:

  ```
  job0 is root job
  ```

  > **Note** Because this query only needs one job (job0), only one line of information is needed.

- **The second part is the dependency between tasks.**

  Command output:

  ```
  In Job job0:
  root Tasks: M1_Stg1, M2_Stg1
  J3_1_2_Stg1 depends on: M1_Stg1, M2_Stg1
  ```

  > **Note**
  >
  > - Job0 contains three tasks, among which M1_Stg1 and M2_Stg1 are executed first, and J3_1_2_Stg1 is executed after the first two tasks are finished.
  > - Naming rules for tasks: MaxCompute provides four task types: MapTask, ReduceTask, JoinTask, and LocalWork. The first letter of a task name indicates the type of the current task (for example, M2Stg1 is a MapTask). The number immediately following the first letter represents the current Task ID, which is unique among all tasks in the current query. The numbers separated by underscores (_) represent the immediate dependencies of the current task. For example, J3_1_2_Stg1 means that the current task (ID 3) is dependent on tasks with ID 1 and ID 2.

- **The third part is the operator structure in the tasks, where each operator string describes the execution semantics of a task.**

  Command output:

```
In Task M1_Stg1:
Data source: yudi_2.src #### "Data source" describes the input content of the current task TS: alia
s: a #### TableScanOperator
RS: order: + #### ReduceSinkOperator keys:
a.value values:
a.key partitions:
a.value
In Task J3_1_2_Stg1:
JOIN: a INNER JOIN b #### JoinOperator
SEL: Abs(UDFToDouble(a._col0)), b._col5 #### SelectOperator FS: output: None #### FileSinkOpera
tor
In Task M2_Stg1:
Data source: yudi_2.src1 TS: alias: b
RS: order: + keys:
b.value values:
b.value partitions:
b.value
```

The meanings of the operators are shown as below.

Operators

Operator	Description
TableScanOperator	Describes the logic of FROM statement blocks in a query statement. The input table name (alias) is displayed in the EXPLAIN results.
SelectOperator	Describes the logic of SELECT statement blocks in a query statement. The columns passed to the next operator, separated by commas, are displayed in the EXPLAIN results. If the result is a reference to a column, it is displayed as < alias >.< column_name >. If the result is an expression, it is displayed as a function, for example, func1(arg1_1, arg1_2, func2(arg2_1, arg2_2)). If the result is a constant, the value is displayed directly.
FilterOperator	Describes the logic of WHERE statement blocks in a query statement. A WHERE condition, which complies with a display rule similar to that of selectOperator, is displayed in the EXPLAIN results.
JoinOperator	Describes the logic of JOIN statement blocks in a query statement. The tables involved in the JOIN operation and the mode of JOIN operation are displayed in the EXPLAIN results.

Operator	Description
GroupByOperator	Describes the logic of the AGGREGATE operation. This structure is displayed if an aggregate function is used in a query. The content of the aggregate function is displayed in the EXPLAIN results.
ReduceSinkOperator	Describes the logic of the data distribution operation between tasks. If the result of the current task is transferred to another task, ReduceSinkOperator must be used to distribute data at the end of the current task. The output sorting method, the distributed keys, values, and columns used to calculate the hash value are displayed in the EXPLAIN results.
FileSinkOperator	Describes the final data storage operation. If there is an INSERT statement block in the query statement, the name of the target table is displayed in the EXPLAIN results.
LimitOperator	Describes the logic of LIMIT statement blocks in a query statement. The limit value is displayed in the EXPLAIN results.
MapjoinOperator	Describes JOIN operations in large tables, similar to JoinOperator.

Note

- If a query is complex and has too many EXPLAIN results, the API restriction is triggered, and incomplete results are displayed. In this case, the query can be split, and the EXPLAIN operation can be performed on each part to show the structure of the job.
- The maximum number of partitions in a query is 10,000. Inputting too many partitions leads to over-length Data source content. To circumvent this limit, you can filter out most partitions by adding a query filter.

1.6.4.6. GROUPING SETS

1.6.4.6.1. Overview

For scenarios where you need to aggregate and analyze data of multiple dimensions, you must execute multiple UNION ALL clauses. For example, you wanted to aggregate column a, aggregate column b, and aggregate columns a and b together. The GROUPING SETS clause is a better choice in such cases.

GROUPING SETS is an extension to the GROUP BY clause in the SELECT statement. You can group results in various ways by using GROUPING SETS without executing multiple SELECT statements. This can produce better execution plans and result in higher performance from the MaxCompute engine.

> **Notice** Many examples in this topic are demonstrated using MaxCompute Studio. We recommend that you install MaxCompute Studio before you proceed with subsequent operations.

1.6.4.6.2. Example

The following example is for your reference.

1. Prepare data.

```
create table requests LIFECYCLE 20 as
select * from values
   (1, 'windows', 'PC', 'Beijing'),
   (2, 'windows', 'PC', 'Shijiazhuang'),
   (3, 'linux', 'Phone', 'Beijing'),
   (4, 'windows', 'PC', 'Beijing'),
   (5, 'ios', 'Phone', 'Shijiazhuang'),
   (6, 'linux', 'PC', 'Beijing'),
   (7, 'windows', 'Phone', 'Shijiazhuang')
as t(id, os, device, city);
```

2. Use GROUPING SETS.

```
SELECT os,device, city ,COUNT(*)
FROM requests
GROUP BY os, device, city GROUPING SETS((os, device), (city), ());
```

A similar output is displayed.

Command output

	os	device	city	_c3
1	ios	Phone	\N	1
2	linux	PC	\N	1
3	linux	Phone	\N	1
4	windows	PC	\N	3
5	windows	Phone	\N	1
6	\N	\N	Beijing	4
7	\N	\N	Shijiazhuang	3
8	\N	\N	\N	7

Note You can also execute multiple SELECT statements to obtain the same result.

```
SELECT NULL, NULL, NULL, COUNT(*)
FROM requests
UNION ALL
SELECT os, device, NULL, COUNT(*)
FROM requests GROUP BY os, device
UNION ALL
SELECT null, null, city, COUNT(*)
FROM requests GROUP BY city;
```

However, the GROUPING SETS method is simpler and more efficient.

Notice Expressions not used in GROUPING SETS use NULL as placeholders. You can execute UNION statements on grouping sets.

1.6.4.6.3. CUBE and ROLLUP

CUBE and ROLLUP are special GROUPING SETS functions. CUBE lists all possible combinations of the specific columns as grouping sets. ROLLUP aggregates data by level to generate grouping sets.

Example

```
GROUP BY CUBE(a, b, c)
-- Equivalent to the following statement:
GROUPING SETS((a,b,c),(a,b),(a,c),(b,c),(a),(b),(c),())

GROUP BY ROLLUP(a, b, c)
-- Equivalent to the following statement:
GROUPING SETS((a,b,c),(a,b),(a), ())

GROUP BY CUBE ( (a, b), (c, d) )
-- Equivalent to the following statement:
GROUPING SETS (
  ( a, b, c, d ),
  ( a, b      ),
  (      c, d ),
  (           )
)

GROUP BY ROLLUP ( a, (b, c), d )
-- Equivalent to the following statement:
```

```
GROUPING SETS (
  ( a, b, c, d ),
  ( a, b, c    ),
  ( a          ),
  (            )
)

GROUP BY a, CUBE (b, c), GROUPING SETS ((d), (e))
-- Equivalent to the following statement:
GROUP BY GROUPING SETS (
  (a, b, c, d), (a, b, c, e),
  (a, b, d),    (a, b, e),
  (a, c, d),    (a, c, e),
  (a, d),       (a, e)
)

GROUP BY grouping sets((b), (c),rollup(a,b,c))
-- Equivalent to the following statement:
GROUP BY GROUPING SETS (
  (b), (c),
  (a,b,c), (a,b), (a), ()
 )
```

1.6.4.6.4. GROUPING and GROUPING_ID

NULL is used as placeholders in grouping sets, but it can also be a value that is manually entered. In the code, however, placeholder NULLs are indistinguishable from value NULLs. The GROUPING function is provided to address this issue.

GROUPING allows you to specify the name of a column as a parameter. If the specified lines are aggregated based on a column whose name is used as a parameter in this function, 0 is returned, indicating that NULL is an entered value. Otherwise, 1 is returned, indicating that NULL is a placeholder.

GROUPING_ID can be used to specify the names of one or more columns as parameters. The GROUPING results in these columns are formed into integers by using BitMap.

Example:

```
SELECT a,b,c ,COUNT(*),
GROUPING(a) ga, GROUPING(b) gb, GROUPING(c) gc, GROUPING_ID(a,b,c) groupingid
FROM VALUES (1,2,3) as t(a,b,c)
GROUP BY CUBE(a,b,c);
```

A similar output is displayed.

Command output

	a	b	c	_c3	ga	gb	gc	groupingid
1	\N	\N	\N	1	1	1	1	7
2	\N	\N	3	1	1	1	0	6
3	\N	2	\N	1	1	0	1	5
4	\N	2	3	1	1	0	0	4
5	1	\N	\N	1	0	1	1	3
6	1	\N	3	1	0	1	0	2
7	1	2	\N	1	0	0	1	1
8	1	2	3	1	0	0	0	0

1.6.4.7. IF statement

MaxCompute SQL supports the IF-ELSE statement.

You can use the IF-ELSE statement to execute SQL scripts with specific conditions. The condition in the IF-ELSE statement can be a standard variable or a scalar subquery that returns only one column value from one row.

The IF statement allows the system to automatically select the execution logic based on the specified conditions. MaxCompute supports the following IF syntax:

```
IF (condition) BEGIN
 statement 1
 statement 2
 ...
END
IF (condition) BEGIN
 statements
END ELSE IF (condition2) BEGIN
 statements
END ELSE BEGIN
 statements
END
```

> **Note** The BEGIN and END conditional clause can be omitted because it contains only one statement, similar to '{ }' in Java.

The IF statement can contain two types of conditions: expressions and scalar subqueries. Both of them are of the BOOLEAN type.

- Expressions: A BOOLEAN-type expression in the IF-ELSE statement determines which branch is executed at the compiling stage. Example:

```
@date := '20190101';
@row TABLE(id STRING); -- Declare the row variable. The type of the row is Table and schema is STR
ING.
IF ( cast(@date as bigint) % 2 == 0 ) BEGIN
@row := SELECT id from src1;
END ELSE BEGIN
@row := SELECT id from src2;
END
INSERT OVERWRITE TABLE dest SELECT * FROM @row;
```

- Scalar subqueries: A BOOLEAN-type scalar subquery in the IF-ELSE statement determines which branch is executed at the running stage. Therefore, you must submit multiple jobs. Example:

```
@i bigint;
@t table(id bigint, value bigint);
IF ((SELECT count(*) FROM src WHERE a = '5') > 1)  BEGIN
@i := 1;
@t := select @i, @i*2;
END ELSE
BEGIN
@i := 2;
@t := select @i, @i*2;
END
select id, value from @t;
```

1.6.5. SELECT TRANSFORM

1.6.5.1. Overview

SELECT TRANSFORM implements features that MaxCompute SQL does not provide. SELECT TRANSFORM allows you to start a specified child process and enter data of a required format into the child process through standard input (stdin). Then, you can parse the standard output (stdout) of the child process to obtain the final output. This process does not require you to compile UDFs.

SELECT TRANSFORM simplifies the reference of script code and supports programming languages such as Java, Python, Shell, and Perl. It is suitable for ad hoc data analysis. MaxCompute Select Transform is fully compatible with Hive syntax, features, and actions, including input/output row format and reader/writer. Most Hive scripts can be added directly to the SELECT TRANSFORM statement. Others can be used after a few changes.

Command syntax:

```
SELECT TRANSFORM(arg1, arg2 ...)
(ROW FORMAT DELIMITED (FIELDS TERMINATED BY field_delimiter (ESCAPED BY character_escape)?)? (LI
NES SEPARATED BY line_separator)? (NULL DEFINED AS null_value)?)?
USING 'unix_command_line'
(RESOURCES 'res_name'  (',' 'res_name') *)?
( AS col1, col2 ...)?
(ROW FORMAT DELIMITED (FIELDS TERMINATED BY field_delimiter (ESCAPED BY character_escape)?)? (LI
NES SEPARATED BY line_separator)? (NULL DEFINED AS null_value)?)?
```

Description:

- SELECT TRANSFORM: The **SELECT TRANSFORM** keyword can be replaced with the **MAP** or **REDUCE** keyword while maintaining the same semantic meaning. However, we recommend that you use**SELECT TRANSFORM** because its syntax is simpler.
- (arg1, arg2 ...): arguments in the TRANSFORM clause. Their format is similar to those of items in the SELECT clause. In the default format, the results of expressions for each argument are combined by using \t after they are implicitly converted into strings. The arguments are then entered into the specified child process.

  > **Note** The default format is configurable. For more information, see **ROW FORMAT**.

- USING: specifies the command used to start a child process. Note the following points about the USING clause.
  - In most MaxCompute SQL statements, the USING clause can only specify resources. However, in the SELECT TRANSFORM statement, the USING clause can specify commands to ensure compatibility with Hive syntax.
  - The format of the USING clause is similar to the syntax of a Shell script. However, a Shell script is not actually expected to start the child process. The child process is created based on the command input. Because of this, a number of Shell functions, such as input and output redirection, pipe, and loop, are unavailable. A Shell script can be used as to start a child process if necessary.
- RESOURCES: specifies the resources that the specified child process can access. You can use one of the following methods to specify resources:
  - Use the RESOURCES clause. Example: `using 'sh foo.sh bar.txt' Resources 'foo.sh','bar.txt'` .
  - Add the `set odps.sql.session.resources=foo.sh,bar.txt;` clause before SQL statements.

    > **Notice** This clause takes effect globally once it is specified. All SELECT TRANSFORM statements will be able to access the resources specified by this clause.

- ROW FORMAT: specifies the input or output format. Two ROW FORMAT clauses are used in the syntax: the first one specifies the input format, and the second one specifies the output format. \t is used to separate columns, \n is used to separate rows, and NULL is represented by \N .

> **Notice**
>
> - For field_delimiter, character_escape, and line_separator, only one character can be accepted. If you specify a string, the first character in the string takes priority over the others.
> - There are a variety of Hive syntaxes to specify formats. MaxCompute supports syntaxes such as inputRecordReader, outputRecordReader, and Serdeinput. To use these formats, you must enable Hive compatibility by adding the `set odps.sql.hive.compatible=true;` clause before SQL statements. If you specify a syntax such as inputRecordReader or outputRecordReader supported by Hive, statements may be executed at lower speeds.

- **AS: specifies output columns.**

> **Note**
>
> - You can specify data types in the AS clause, as in as(col1:bigint, col2:boolean). By default, strings are returned if you do not specify data types, as in as(col1, col2).
> - The output is obtained by parsing the stdout of the child process. If the specified data types do not include STRING, the system implicitly calls the CAST function. Runtime exceptions may occur when the CAST function is called.
> - You cannot specify data types for only some of the columns, as in as(col1, col2:bigint).
> - If you skip the AS clause, the field preceding the first \t in the stdout is a key, and all the following parts are a value. This is equivalent to as(key, value).

1.6.5.2. SELECT TRANSFORM examples

1.6.5.2.1. Call Shell scripts

In this example, a Shell script is used to generate 50 lines of data starting from 1 to 50. The output of the data field is as follows:

```
SELECT  TRANSFORM(script) USING 'sh' AS (data)
FROM   (
       SELECT  'for i in `seq 1 50`; do echo $i; done' AS script
    ) t
;
```

The Shell commands are used as the input of the TRANSFORM clause.

> **Note** In addition to language extensions, SELECT TRANSFORM also provides simple features of AWK, Python, Perl, and Shell to compile scripts in commands. You do not need to compile script files or upload resources separately.

You can upload script files for complex cases, as in the following example Python script call.

1.6.5.2.2. Call Python scripts

This topic provides an example of how to use SELECT TRANSFORM to call Python scripts.

1. Compile a Python script file. In this example, the file name is myplus.py.

```
#! /usr/bin/env python
import sys
line = sys.stdin.readline()
while line:
  token = line.split('\t')
  if (token[0] == '\\N') or (token[1] == '\\N'):
    print '\\N'
  else:
    print int(token[0]) + int(token[1])
  line = sys.stdin.readline()
```

2. Add the Python script file as a resource to MaxCompute.

```
add py ./myplus.py  -f;
```

> **Note** You can also add resources from the DataWorks console.

3. Execute the SELECT TRANSFORM statement to call the resource.

```
Create table testdata(c1 bigint,c2 bigint); -- Create a test table.
insert into Table testdata values (1,4),(2,5),(3,6); -- Insert test data into the test table.
-- Execute the SELECT TRANSFORM statement:
SELECT
TRANSFORM (testdata.c1, testdata.c2)
USING 'python myplus.py'resources 'myplus.py'
AS (result bigint)
FROM testdata;
-- Or
set odps.sql.session.resources=myplus.py;
SELECT
TRANSFORM (testdata.c1, testdata.c2)
USING 'python myplus.py'
AS (result bigint)
FROM testdata;
```

4. A similar output is displayed:

```
+-----+
| cnt |
+-----+
| 5   |
| 7   |
| 9   |
+-----+
```

Python scripts are not subject to any format requirements and do not require a Python framework to be run in MaxCompute. In MaxCompute, Python commands can be used as the input of the TRANSFORM clause. For example, you can call Shell scripts by running Python commands.

```
SELECT TRANSFORM('for i in xrange(1, 50):  print i;') USING 'python' AS (data);
```

1.6.5.2.3. Call Java scripts

Java scripts are called in a similar manner to Python scripts. In this example, you need to compile a Java script file, export it as a JAR package, and then run the add command to add the JAR package as a resource to MaxCompute. The resource will be called by using SELECT TRANSFORM.

1. Compile a Java script file and export it as a JAR package. In this example, the name of the JAR package is Sum.jar.

```
package com.aliyun.odps.test;
import java.util.Scanner;
public class Sum {
 public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  while (sc.hasNext()) {
   String s = sc.nextLine();
   String[] tokens = s.split("\t");
   if (tokens.length < 2) {
    throw new RuntimeException("illegal input");
   }
   if (tokens[0].equals("\\N") || tokens[1].equals("\\N")) {
    System.out.println("\\N");
   }
   System.out.println(Long.parseLong(tokens[0]) + Long.parseLong(tokens[1]));
  }
 }
}
```

2. Add the JAR package as a resource to MaxCompute.

```
add jar . /Sum.jar -f;
```

3. Execute the SELECT TRANSFORM statement to call the resource.

```
Create table testdata(c1 bigint,c2 bigint); -- Create a test table.
insert into Table testdata values (1,4),(2,5),(3,6); -- Insert test data into the test table.
-- Execute the SELECT TRANSFORM statement:
SELECT TRANSFORM(testdata.c1, testdata.c2)
  USING 'java -cp Sum.jar com.aliyun.odps.test.Sum' resources 'Sum.jar'
from testdata;
-- Or
set odps.sql.session.resources=Sum.jar;
SELECT TRANSFORM(testdata.c1, testdata.c2)
  USING 'java -cp Sum.jar com.aliyun.odps.test.Sum'
FROM testdata;
```

4. A similar output is displayed:

```
+------+
| cnt |
+------+
| 5   |
| 7   |
| 9   |
+------+
```

You can use the preceding method to run most Java utilities.

Although UDTF frameworks are provided for Java and Python, it is easier to compile code by using SELECT TRANSFORM. SELECT TRANSFORM is a simpler process because it is not subject to any format requirements and can be called offline. The paths for Java and Python offline scripts can be obtained from the JAVA_HOME and PYTHON_HOME environment variables.

1.6.5.2.4. Call scripts of other languages

In addition to language extensions, SELECT TRANSFORM also supports commonly used Unix command and script interpreters, such as AWK and Perl.

An example of calling AWK:

```
SELECT TRANSFORM(*) USING "awk '//{print $2}'" as (data) from testdata;
```

An example of calling Perl:

```
SELECT TRANSFORM (testdata.c1, testdata.c2) USING "perl -e 'while($input = <STDIN>){print $input;}'" F
ROM testdata;
```

> **Notice** PHP and Ruby are not deployed in the MaxCompute cluster and cannot be called.

1.6.5.2.5. Call scripts in series

SELECT TRANSFORM allows you to call scripts in series. For example, you can use DISTRIBUTE BY and SORT BY to pre-process data.

```
SELECT TRANSFORM(key, value) USING 'cmd2' from
(
  SELECT TRANSFORM(*) USINg 'cmd1' from
  (
    SELECt * FROM data distribute by col2 sort by col1
  ) t distribute by key sort by value
) t2;
```

More often, you can use either the map or reduce keywords to produce the same results.

```
@a := select * from data distribute by col2 sort by col1;
@b := map * using 'cmd1' distribute by col1 sort by col2 from @a;
reduce * using 'cmd2' from @b;
```

1.6.5.3. Performance advantages

The performance of SELECT TRANSFORM and UDTF varies depending on the specific scenario. In general, SELECT TRANSFORM performs better. However, UDTF performs better as the volume of data increases. Because the development of transform is easier, SELECT TRANSFORM is more suitable for ad hoc data analysis.

The advantages of UDTFs and SELECT TRANSFORM are listed in the following sections.

Advantages of UDTFs

- Output and input follow specified data types and do not require conversion.
- Processes are not suspended if the operating system pipe is empty or fully occupied. The operating system pipe has a 4 KB buffer.
- Constant parameters do not need to be transmitted.

Advantages of SELECT TRANSFORM

- Supports child and parent processes and can utilize multiple server cores when high CPU usage and low throughput is needed.
- Calls underlying systems to read and write data to be transmitted, giving it a higher performance than Java.
- Supports tools such as AWK and can run native code.

1.6.6. UNION, INTERSECT, and EXCEPT

This topic describes SQL syntax, descriptions and examples of UNOIN ALL, UNION DISTINCT, INTERSECT ALL, INTERSECT DISTINCT, EXCEPT ALL, and EXCEPT DISTINCT.

Syntax:

```
select_statement UNION ALL select_statement;
select_statement UNION [DISTINCT] select_statement;
select_statement INTERSECT ALL select_statement;
select_statement INTERSECT [DISTINCT] select_statement;
select_statement EXCEPT ALL select_statement;
select_statement EXCEPT [DISTINCT] select_statement;
select_statement MINUS ALL select_statement;
select_statement MINUS [DISTINCT] select_statement;
```

Purpose: It is used to return the union of two data sets, the intersection of two data sets, or the complement of the second dataset in the first dataset.

Description:

- UNION: returns the union of two datasets. It combines the two datasets into one dataset.
- INTERSECT: returns the intersection of two datasets. It outputs the records contained in both datasets.
- EXCEPT: returns the complement of the second dataset in the first dataset. It outputs the records that are contained in the first dataset, but not in the second dataset.
- MINUS: equivalent to EXCEPT.

Examples:

- **UNOIN ALL example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4) t(a, b)
 UNION ALL
SELECT * FROM VALUES (1, 2), (1, 4) t(a, b);
```

Returned result: two datasets are combined.

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 1          | 4          |
| 1          | 2          |
| 1          | 2          |
| 3          | 4          |
+------------+------------+
```

- **UNION DISTINCT example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4) t(a, b)
UNION
SELECT * FROM VALUES (1, 2), (1, 4) t(a, b);
```

Returned result: equivalent to `SELECT DISTINCT * FROM (< the result of UNOIN ALL >) t;` .

```
+------------+------------+
| a       | b       |
+------------+------------+
| 1       | 2       |
| 1       | 4       |
| 3       | 4       |
+------------+------------+
```

- **INTERSECT ALL example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (5, 6) t(a, b)
INTERSECT ALL
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (5, 7) t(a, b);
```

Returned result: deduplication is skipped in INTERSECT ALL. It seems that there is a hidden serial number behind the same row and each row can be displayed separately.

```
+------------+------------+
| a       | b       |
+------------+------------+
| 1       | 2       |
| 1       | 2       |
| 3       | 4       |
+------------+------------+
```

- **INTERSECT DISTINCT example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (5, 6) t(a, b)
INTERSECT
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (5, 7) t(a, b);
```

Returned result: `SELECT DISTINCT * FROM (< the result of INTERSECT ALL >) t;` .

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 3          | 4          |
+------------+------------+
```

- **EXCEPT ALL example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
EXCEPT ALL
SELECT * FROM VALUES (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
```

Returned result: deduplication is skipped in EXCEPT ALL. There is a hidden serial number behind the same row and each row can be displayed separately.

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 1          | 2          |
| 3          | 4          |
| 7          | 8          |
+------------+------------+
```

- **EXCEPT DISTINCT example:**

```
SELECT * FROM VALUES (1, 2), (1, 2), (3, 4), (3, 4), (5, 6), (7, 8) t(a, b)
EXCEPT
SELECT * FROM VALUES (3, 4), (5, 6), (5, 6), (9, 10) t(a, b);
```

Returned result: equivalent to `Select distinct * FROM left_branch limit t all select distinct * FROM right_branch;` .

```
+------------+------------+
| a          | b          |
+------------+------------+
| 1          | 2          |
| 7          | 8          |
+------------+------------+
```

Note

- Sorting may be skipped in the preceding operations.
- The left and right branches in the preceding operations must have the same number of columns. In addition, if data types in the left and right branches are not consistent, they may be implicitly converted. Due to compatibility issues, implicit conversion is not carried out between STRING and no-STRING types for the preceding operations.
- Up to 256 branches are allowed in the preceding operations. An error is returned if more branches are used.
- If the UNION statement is followed by the CLUSTER BY, DISTRIBUTE BY, SORT BY, ORDER BY or LIMIT clause and you add `set odps.sql.type.system.odps2=false;`, the SET statement is applicable to the last `select_statement;` of the UNION statement. If you add `set odps.sql.type.system.odps2=true;`, the SET statement is applicable to all select_statements of the UNION statement. Example:

  ```
  set odps.sql.type.system.odps2=true;
  SELECT explode(array(3, 1)) AS (a) UNION ALL SELECT explode(array(0, 4, 2)) AS (a) ORDER BY a LIMIT 3;
  ```

  Returned result:

  ```
  +------+
  | a    |
  +------+
  | 0    |
  | 1    |
  | 2    |
  +------+
  ```

1.6.7. Built-in functions

1.6.7.1. Mathematical functions

1.6.7.1.1. ABS

This topic describes the ABS function.

Function declaration:

```
double abs(double number)
bigint abs(bigint number)
decimal abs(decimal number)
```

Purpose: It is used to return absolute values.

Description:

number: double, bigint or decimal type. When the input is of the bigint type, a value of the bigint type is returned; when the input is of the double type, a value of the double type is returned. If the input is of the string type, it is implicitly converted into a value of the double type before this computation. If the input is of another type, an error is returned.

Returned value: double, bigint, or decimal type, depending on the type of the input. If the input is NULL, NULL is returned.

> **Note** When the input is of the bigint type and is out of the maximum range of the bigint type, the returned value is of the double type. In this case, the precision may be diminished.

Example:

```
abs(null) = null
abs(-1) = 1
abs(-1.2) = 1.2
abs("-2") = 2.0
abs(12232083745629837659238745692 3748) = 1.2232083745629837e32
```

The following example shows the usage of a complete ABS function in SQL. Other built-in functions (except window functions and aggregation functions) are in similar usage to this function and are not shown here.

```
select abs(id) from tbl1;
-- Take the absolute value of the id field in tbl1.
```

1.6.7.1.2. ACOS

Function declaration:

```
double acos(double number)
decimal acos(decimal number)
```

Purpose: It is used to calculate the arccosine of a number.

Description:

number: double or decimal type. Value range: -1 to 1. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. Value range: 0 to π. If number is NULL, NULL is returned.

Example:

```
acos("0.87") = 0.5155940062460905
acos(0) = 1.5707963267948966
```

1.6.7.1.3. ASIN

Function declaration:

```
double asin(double number)
decimal asin(DECIMAL number)
```

Purpose: It is used to calculate the arcsine of a number.

Description:

number: double or decimal type. Value range: -1 to 1. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. Value range: -π/2 to π/2. If number is NULL, NULL is returned.

Example:

```
asin(1) = 1.5707963267948966
asin(-1) = -1.5707963267948966
```

1.6.7.1.4. ATAN

Function declaration:

```
double atan(double number)
```

Purpose: It is used to calculate the arctangent of a number.

Description:

number: double type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double type. Value range: -π/2 to π/2. If number is NULL, NULL is returned.

Example:

```
atan(1) = 0.7853981633974483;
atan(-1) = -0.7853981633974483
```

1.6.7.1.5. CEIL

Command syntax:

```
bigint ceil(double value)
bigint ceil(decimal value)
```

Purpose: It is uesed to return the smallest integer that is equal to or greater than the input value.

Description:

value: double or decimal. If the value is of the string or bigint type, it is implicitly converted to the double type. For all other input types, an error is returned.

Returned value: bigint type. If the input is NULL, NULL is returned.

Example:

```
ceil(1.1) = 2
ceil(-1.1) = -1
```

1.6.7.1.6. CONV

Command syntax:

```
string conv(string input, bigint from_base, bigint to_base)
```

Purpose: It is ued to convert a number from one numeric base number system to another.

Description:

- input: an integer of the string type to be converted. It accepts values of the bigint and double types by means of implicit conversion.
- from_base, to_base: a number system value in decimal form. Value range: 2, 8, 10, and 16. It accepts values of the string and double types by means of implicit conversion.

Returned value: string type. If any input is NULL, NULL is returned. The conversion process runs at a 64-bit precision. An error is returned when overflow occurs. If the input is a negative value (beginning with '-'), an error is returned. If the input is a decimal, it is converted to an integer before hex conversion. The decimal part is left out.

Example:

```
conv('1100', 2, 10) = '12'
conv('1100', 2, 16) = 'c'
conv('ab', 16, 10) = '171'
conv('ab', 16, 16) = 'ab'
```

1.6.7.1.7. COS

Command syntax:

```
double cos(double number)
decimal cos(decimal number)
```

Purpose: It is used to return the cosine of a number. The input must be a radian value.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted to a value of the double type. For all other input types, an error is returned.

Returned value: double or decimal type. If the input is NULL, NULL is returned.

Example:

```
cos(3.1415926/2) = 2.6794896585028633e-8
cos(3.1415926) = -0.9999999999999986
```

1.6.7.1.8. COSH

Command syntax:

```
double cosh(double number)
decimal cosh(decimal number)
```

Purpose: It is used to return the hyperbolic cosine of a number.

Description:

number: double or decimal. If the input is of the string or bigint type, it is implicitly converted to a value of the double type. For all other input types, an error is returned.

Returned value: double or decimal. If the input is NULL, NULL is returned.

1.6.7.1.9. COT

Function declaration:

```
double cot(double number)
decimal cot(decimal number)
```

Purpose: It is used to return the cotangent of a number. The input must be a radian value.

Description:

number: double or decimal. If the input is of the string or bigint type, it is implicitly converted a value of the double type. For all other input types, an error is returned.

Returned value: double or decimal type. If the input is NULL, NULL is returned.

1.6.7.1.10. EXP

Function declaration:

```
double exp(double number)
decimal exp(decimal number)
```

Purpose: It is used to return the exponent value of number.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.11. FLOOR

Function declaration:

```
bigint floor(double number)
bigint floor(decimal number)
```

Purpose: It is used to return the round-down integer that is less than or equal to number.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: bigint type. If number is NULL, NULL is returned.

Example:

```
floor(1.2) = 1
floor(1.9) = 1
floor(0.1) = 0
floor(-1.2) = -2
floor(-0.1) = -1
floor(0.0) = 0
floor(-0.0) = 0
```

1.6.7.1.12. LN

Function declaration:

```
double ln(double number)
decimal ln(decimal number)
```

Purpose: It is used to return the natural logarithm of a number.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. If the input is NULL, negative, or zero, NULL is returned.

1.6.7.1.13. LOG

Function declaration:

```
double log(double base, double x)
decimal log(decimal base, DECIMAL x)
```

Purpose: It is used to return the logarithm of x to base.

Description:

- base: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.
- x: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: logarithm value of the double or decimal type. If either base or x is NULL, negative, or zero, NULL is returned. If base is 1 (which leads to division by zero), NULL is returned.

1.6.7.1.14. POW

Command syntax:

```
double pow(double x, double y)
decimal pow(decimal x, decimal y)
```

Purpose: It is used to return the yth power of x, that is, x^y.

Description:

- x: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.
- y: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. If x or y is NULL, NULL is returned.

1.6.7.1.15. RAND

Command syntax:

```
double rand(bigint seed)
```

Purpose: It is used to return a random number of the double type from 0 to 1 based on the seed.

Description:

Seed: optional, bigint type. It is the seed of a random number, and determines the start value of the random number sequence.

Returned value: double type.

Example:

```
select rand() from dual;
select rand(1) from dual;
```

1.6.7.1.16. ROUND

Function declaration:

```
double round(double number, [bigint decimal_places])
decimal round(decimal number, [bigint decimal_places])
```

Purpose: It is used to return a number rounded to the specified decimal place.

Description:

- number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. If the input is of another type, an error is returned.
- decimal_place: a constant of the bigint type. It indicates the specified decimal place to which the result is to be rounded off. For all other input types, an error is returned. If it is omitted, the number is rounded to the ones place. The default value is 0.

Returned value: double or decimal type. If number or decimal_places is NULL, NULL is returned.

> **Note** decimal_places can be negative. Negative numbers are counted from the decimal point to left and the decimal part is left out; if the value of decimal_places is greater than the length of the integer part, 0 is returned.

Example:

```
round(125.315) = 125.0
round(125.315, 0) = 125.0
Round (125.315, 1) = 125.3
round(125.315, 2) = 125.32
round(125.315, 3) = 125.315
round(-125.315, 2) = -125.32
round(123.345, -2) = 100.0
round(null) = null
round(123.345, 4) = 123.345
round(123.345, -4) = 0.0
```

1.6.7.1.17. SIN

Function declaration:

```
double sin(double number)
decimal sin(decimal number)
```

Purpose: It is used to return the sine of a number. The input must be a radian value.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.18. SINH

Function declaration:

```
double sinh(double number)
decimal sinh(decimal number)
```

Purpose: It is used to return the hyperbolic sine of a number.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.19. SQRT

Function declaration:

```
double sqrt(double number)
decimal sqrt(decimal number)
```

Purpose: It is used to return the square root of a number.

Description:

number: double or decimal type. It must be greater than 0. If it is less than 0, an error is returned. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.20. TAN

Function declaration:

```
double tan(double number)
decimal tan(decimal number)
```

Purpose: It is used to return the tangent of a number. The input must be a radian value.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.21. TANH

Function declaration:

```
double tanh(double number)
decimal tanh(decimal number)
```

Purpose: It is used to return the hyperbolic tangent of a number.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned.

Returned value: double or decimal type. If number is NULL, NULL is returned.

1.6.7.1.22. TRUNC

Function declaration:

```
double trunc(double number[, bigint decimal_places])
decimal trunc(decimal number[, bigint decimal_places])
```

Purpose: It is used to truncate 'number' to the specified decimal place.

Description:

- number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned.
- decimal_places: a constant of the bigint type. It indicates the decimal place to which a number is to be truncated. Numbers of other types are implicitly converted into values of the bigint type. If it is omitted, the result is truncated to the ones place by default.

Returned value: double or decimal type. If number or decimal_places is NULL, NULL is returned.

> **Note**
>
> - The truncated part is supplemented with 0.
> - decimal_places can be negative. Negative numbers are truncated from the decimal point to the left and the decimal part is left out. If the value of decimal_places is greater than the length of the integer part, 0 is returned.

Example:

```
trunc(125.815) = 125.0
trunc(125.815, 0) =125.0
trunc(125.815, 1) = 125.80000000000001
trunc(125.815, 2) = 125.81
trunc(125.815, 3) = 125.815
trunc(-125.815, 2) = -125.81
trunc(125.815, -1) = 120.0
trunc(125.815, -2) = 100.0
trunc(125.815, -3) = 0.0
trunc(123.345, 4) = 123.345
trunc(123.345, -4) = 0.0
```

1.6.7.1.23. Additional mathematical functions

MaxCompute 2.0 provides additional mathematical functions. You must add the following SET statement before SQL statements contained in the UNHEX function:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements of the new functions simultaneously.

The mathematical functions described in subsequent topics are new in MaxCompute 2.0.

1.6.7.1.24. LOG2

Function declaration:

```
Double log2(DOUBLE number)
Double log2(DECIMAL number)
```

Purpose: It is used to return the logarithm of number to base 2.

Description:

number: double or decimal type.

Returned value: double type. If the input is 0 or NULL, NULL is returned.

Example:

```
log2(null) = null
log2(0) = null
log2(8) = 3.0
```

1.6.7.1.25. LOG10

Function declaration:

```
Double log10(Double number)
Double log10(Decimal number)
```

Purpose: It is used to return the logarithm of number to base 10.

Description:

number: double or decimal type.

Returned value: double type. If the input is 0 or NULL, NULL is returned.

Example:

```
log10(null) = null
log10(0) = null
log10(8) = 0.9030899869919435
log10('abc') = null
```

1.6.7.1.26. BIN

Command syntax:

```
string bin(bigint number)
```

Purpose: It is used to return the binary format of a number.

Description:

number: bigint.

Returned value: string type. If the input is 0, 0 is returned. If the input is NULL, NULL is returned.

Example:

```
bin(0) = '0'
bin(null) = 'null'
bin(12) = '1100'
```

1.6.7.1.27. HEX

Function declaration:

```
STRING hex(BIGINT number)
STRING hex(STRING number)
STRING hex(BINARY number)
```

Purpose: It is used to convert an integer or character into hexadecimal format.

Description:

number: If this value is of the bigint type, the hexadecimal format of the number is returned. If this value is of the string type, the hexadecimal value of the string is returned.

Returned value: string type. If the input is 0, 0 is returned. If the input is NULL, NULL is returned.

Example:

```
hex(0) = '0'
hex('abc') = '616263'
hex(17) = '11'
hex('17') = '3137'
hex(null) = 'null'
```

1.6.7.1.28. UNHEX

Function declaration:

```
BINARY unhex(STRING number)
```

Purpose: It is used to return the regular character string represented in the hexadecimal format.

Description:

number: a hexadecimal string.

Returned value: binary type. If the input is 0, a failure is returned. If the input is NULL, NULL is returned.

Example:

```
unhex('616263') = 'abc'
unhex(616263) = 'abc'
```

1.6.7.1.29. RADIANS

Command syntax:

```
double radians(double number)
```

Purpose: It is used to convert degrees into radians.

Description:

number: double type

Returned value: double type. If the input is NULL, NULL is returned.

Example:

```
radians(90) = 1.5707963267948966
radians(0) = 0.0
radians(null) = null
```

1.6.7.1.30. DEGREES

Function declaration:

```
DOUBLE degrees(DOUBLE number)
DOUBLE degrees(DECIMAL number)
```

Purpose: It is used to convert radians into degrees.

Description:

number: double or decimal type.

Returned value: double type. If the input is NULL, NULL is returned.

Example:

```
degrees(1.5707963267948966) = 90.0
degrees(0) = 0.0
degrees(null) = null
```

1.6.7.1.31. SIGN

Function declaration:

```
DOUBLE sign(DOUBLE number)
DOUBLE sign(DECIMAL number)
```

Purpose: It is used to indicate the sign of the input data. 1.0 indicates positive and -1.0 indicates negative. 0.0 indicates 0.

Description:

number: double or decimal type.

Returned value: double type. If the input is 0, 0.0 is returned. If the input is NULL, NULL is returned.

Example:

```
sign(-2.5) = -1.0
sign(2.5) = 1.0
sign(0) = 0.0
sign(null) = null
```

1.6.7.1.32. E

Function declaration:

```
DOUBLE e()
```

Purpose: It is used to return the value of e (Euler's number).

Returned value: double type.

Example:

```
e() = 2.718281828459045
```

1.6.7.1.33. PI

Function declaration:

```
DOUBLE pi()
```

Purpose: It is used to return the value of π.

Returned value: double type.

Example:

```
pi() = 3.141592653589793
```

1.6.7.1.34. FACTORIAL

Function declaration:

```
BIGINT factorial(INT number)
```

Purpose: It is used to return the factorial of number.

Description:

number: int type. Value range: 0 to 20.

Returned value: bigint type. If the input is 0, 1 is returned. If the input is NULL or any value outside the range of 0 to 20, NULL is returned.

Example:

```
factorial(5) = 120 --5! = 5*4*3*2*1 = 120
```

1.6.7.1.35. CBRT

Command syntax:

```
double cbrt(double number)
```

Purpose: It is used to return the cube root of a number.

Description:

number: double type.

Returned value: double type. If the input is NULL, NULL is returned.

Example:

```
cbrt(8) = 2
cbrt(null) = null
```

1.6.7.1.36. SHIFTLEFT

Function declaration:

```
INT shiftleft(TINYINT|SMALLINT|INT number1, INT number2)
BIGINT shiftleft(BIGINT number1, INT number2)
```

Purpose: It is used to shift left a value by a given number of places (<<).

Description:

- number1: an integer of the tinyint, smallint, int, or bigint type.
- number2: an integer of the int type.

Returned value: int or bigint type.

Example:

```
shiftleft(1,2) = 4
-- Shift left the binary value of 1 by two places (1<<2, 0001 changed to 0100)
shiftleft(4,3) = 32
-- Shift left the binary value of 4 by three places (4<<3, 0100 changed to 100000)
```

1.6.7.1.37. SHIFTRIGHT

Function declaration:

```
INT shiftright(TINYINT|SMALLINT|INT number1, INT number2)
BIGINT shiftright(BIGINT number1, INT number2)
```

Purpose: It is used to shift right a value by a given number of places (>>).

Description:

- number1: an integer of the tinyint, smallint, int, or bigint type.

- number2: an integer of the int type.

Returned value: int or bigint type.

Example:

```
shiftright(4,2) = 1
-- Shift right the unsigned binary value of 4 by two places (4>>2, 0100 changed to 0001)
shiftright(32,3) = 4
-- Shift right the unsigned binary value of 32 by two places (32>>3, 100000 changed to 0100)
```

1.6.7.1.38. SHIFTRIGHTUNSIGNED

Function declaration:

```
INT shiftrightunsigned(TINYINT|SMALLINT|INT number1, INT number2)
BIGINT shiftrightunsigned(BIGINT number1, INT number2)
```

Purpose: It is used to shift right an unsigned value by a given number of places (>>>).

Description:

- number1: an integer of the tinyint, smallint, int, or bigint type.
- number2: an integer of the int type.

Returned value: int or bigint type.

Example:

```
shiftrightunsigned(8,2) = 2
-- In this example, shift right the unsigned binary value of 8 (1000 in binary) by two places and return 2
(0010 in binary).
shiftrightunsigned(-14,2) = 1073741820
-- Shift right the unsigned binary value of -14 by two places (-14>>>2, 11111111 11111111 11111111 111
10010 changed to 00111111 11111111 11111111 11111100)
```

1.6.7.2. String processing functions

1.6.7.2.1. CHAR_MATCHCOUNT

Command syntax:

```
bigint char_matchcount(string str1, string str2)
```

Purpose: It is used to return the number of characters in str1 that appear in str2 (repeated characters are not counted).

Description:

str1 and str2: string type. Both must be valid UTF-8 strings. If invalid characters are found during matching, a negative value is returned.

Returned value: bigint type. If any input is NULL, NULL is returned.

Example:

```
char_matchcount('abd', 'aabc') = 2
-- The a and b characters in str1 appear in str2.
```

1.6.7.2.2. CHR

Command syntax:

```
string chr(bigint ascii)
```

Purpose: It is used to convert an ASCII code into the corresponding character.

Description:

ascii: ASCII value of the bigint type. If the input is of the string, double, or decimal type, it is implicitly converted into a value of the bigint type before this computation. If the input is of another type, an error is returned.

Returned value: string type. The parameter value range is from 0 to 255. A value out of range will cause an error. If the input is NULL, NULL is returned.

1.6.7.2.3. CONCAT

Command syntax:

```
string concat(string a, string b...)
```

Purpose: It is used to join input strings into a single string.

Description:

a, b...: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type. For all other input types, an error is returned.

Returned value: string type. If there is no input or if any input is NULL, NULL is returned.

Example:

```
concat('ab', 'c') = 'abc'
concat() = null
concat('a', null, 'b') = null
```

1.6.7.2.4. INSTR

Function declaration:

```
bigint instr(string str1, string str2[, bigint start_position[, bigint nth_appearance]])
```

Purpose: It is used to calculate the position of substring str2 in string str1.

Description:

- str1: string type. It indicates a string to be searched. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.
- str2: string type. It indicates a substring to be searched out. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.
- start_position: bigint type. If it is of another type, an error is returned. It indicates which character in str1 the search will start with. The default start position is the first character, marked as 1.
- nth_appearance: bigint type. If it is greater than 0, it indicates the position where the substring matches the string for the nth_appearance time. If it is of another type or if it is less than or equal to 0, an error is returned.

Returned value: bigint type.

> **Note**
>
> - If str2 is not found in str1, 0 is returned.
> - If any input is NULL, NULL is returned.
> - If str2 is NULL, the matching will always be successful. Therefore, 1 is returned for instr('abc', '').

Example:

```
instr('Tech on the net', 'e') = 2
instr('Tech on the net', 'e', 1, 1) = 2
instr('Tech on the net', 'e', 1, 2) = 11
instr('Tech on the net', 'e', 1, 3) = 14
```

1.6.7.2.5. IS_ENCODING

Function declaration:

```
boolean is_encoding(string str, string from_encoding, string to_encoding)
```

Purpose: It is used to determine whether an input string can be converted from a specified character set (from_encoding) to another character set (to_encoding). It can be used to determine whether the input is garbled. from_encoding is usually set to utf-8, and to_encoding is set to gbk.

Description:

- str: string type. If the input is NULL, NULL is returned. Null is considered to belong to any

character set.

- from_encoding, to_encoding: string type. They indicate the source and the destination character sets respectively. If the input is NULL, NULL is returned.

Returned value: boolean type. If a string is converted successfully, true is returned. Otherwise, false is returned.

Example:

```
is_encoding('test', 'utf-8', 'gbk') = true
is_encoding('test', 'utf-8', 'gbk') = true
-- These two traditional Chinese characters are in GBK stock in China.
is_encoding('test', 'utf-8', 'gb2312') = false
-- The grapheme inventory of 'GB2312' does not contain these two Chinese characters.
```

1.6.7.2.6. KEYVALUE

Function declaration:

```
KEYVALUE(STRING srcStr, STRING split1, STRING split2, STRING key)
KEYVALUE(STRING srcStr, STRING key) //split1 = ";", split2 = ":"
```

Purpose: It is used to split the source string into key-value pairs by split1, separate key-value pairs by split2, and return the value of the corresponding key.

Description:

- srcStr: the source string to be split.
- key: string type. After the source string is split by 'split1' and 'split2', return the corresponding value according to the specification of the 'key' value.
- split1 and split2: strings used as separators. The source string is split by the two separators. If these two parameters are not specified in the expression, split1 is a semicolon (;) and split2 is a colon (:) by default. If a string that has been split by split1 has multiple split2 values, the returned result is undefined.

Returned value: string type.

- If 'split1' or 'split2' is NULL, return NULL.
- If 'scrStr' and 'key' are NULL or if there is no matched 'key', return NULL.
- If multiple 'key-value' matches, return the value corresponding to the first matched key.

Example:

```
keyvalue('0:1\;1:2', 1) = '2'
-- The source string is "0:1\;1:2". Because split1 and split2 are not specified, split1 is a semicolon (;) a
nd split2 is a colon (:) by default. After split1 split, the key-value pair is:
0:1\,1:2
After split2 split, it becomes:
0 1/
1 2
Returns the value(2) of the key corresponding to 1.
keyvalue("\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;market:shoes\;in
stPayAmount:0\;", "\;",":","tf") = "21910"
-- The source string is "\;decreaseStore:1\;xcard:1\;isB2C:1\;tf:21910\;cart:1\;shipping:2\;pf:0\;mar
ket:shoes\;instPayAmount:0\;". After the source string is split by split1 "\;", the key-value pairs are a
s follows:
decreaseStore:1, xcard:1, isB2C:1, tf:21910, cart:1, shipping:2, pf:0, market:shoes, instPayAmount:0
 If split2 is ":", after split it becomes:
decreaseStore 1
xcard 1
isB2C 1
tf 21910
cart 1
shipping 2
pf 0
market shoes
instPayAmount 0
For the key parameter whose value is "tf", the returned value of the corresponding value parameter is
21910.
```

1.6.7.2.7. LENGTH

Function declaration:

```
bigint length(string str)
```

Purpose: It is used to return the length of a string.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: bigint type. If a string is NULL, NULL is returned. If a string is not UTF-8 encoded, -1 is returned.

Example:

```
length('hi! China') = 6
```

1.6.7.2.8. LENGTHB

Function declaration:

```
bigint lengthb(string str)
```

Purpose: It is used to return the length of a string. Unit: byte.

Description:

str: string type. If the input is of the bigint, double, decimal, or datetime type, it is implicitly converted into a value of the string type before this computation. If the input is of another type, an error is returned.

Returned value: bigint type. If the input is NULL, NULL is returned.

Example:

```
lengthb('hi! china') = 10
```

1.6.7.2.9. MD5

Function declaration:

```
string md5(string value)
```

Purpose: It is used to calculate the MD5 value of the input string value.

Description:

value: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. If the input is of another type, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

1.6.7.2.10. PARSE_URL

Function declaration:

```
STRING PARSE_URL(STRING url, STRING part[,STRING key])
```

Purpose: It is used to parse a URL and extract information by key.

Description:

- If URL or part is NULL, NULL is returned. If URL is invalid, an error is returned.
- part: string type. It supports HOST, PATH, QUERY, REF, PROTOCOL, AUTHORITY, FILE, and USERINFO, and is case insensitive. If it is none of the preceding values, an error is returned.
- If part is QUERY, the value in query string that corresponds to the key value is extracted.

Otherwise, the parameter key is ignored.

Returned value: string type.

Example:

```
url = file://username:password@example.com:8042/over/there/index.dtb? type=animal&name=narwh
al#nose
parse_url('url', 'HOST') = "example.com"
parse_url('url', 'PATH') = "/over/there/index.dtb"
parse_url('url', 'QUERY') = "type=animal&name=narwhal"
parse_url('url', 'QUERY', 'name') = "narwhal"
parse_url('url', 'REF') = "nose"
parse_url('url', 'PROTOCOL') = "file"
parse_url('url', 'AUTHORITY') = "username:password@example.com:8042"
parse_url('url', 'FILE') = "/over/there/index.dtb? type=animal&name=narwhal"
parse_url('url', 'USERINFO') = "username:password"
```

1.6.7.2.11. REGEXP_EXTRACT

Command syntax:

```
string regexp_extract(string source, string pattern[, bigint occurrence])
```

Purpose: It is used to return part of the source string that matches the regular expression and the occurrence of the matches.

Description:

- source: string type. It indicates a string to be searched.
- pattern: string type. If pattern is NULL or if there is no specified group in pattern, an error is returned.
- occurrence: bigint type. It must be a number that is greater than or equal to 0. Otherwise, an error is returned. The default value is 1 if it is not specified. If it is 0, a substring which meets all pattern requirements is returned.

Returned value: string type. If any input is NULL, NULL is returned.

Example:

```
regexp_extract('foothebar', 'foo(. *?)( bar)', 1) = the
regexp_extract('foothebar', 'foo(. *?)( bar)', 2) = bar
regexp_extract('foothebar', 'foo(. *?)( bar)', 0) = foothebar
regexp_extract('8d99d8', '8d(\\d+)d8') = 99
-- If the regular expression is submitted at the MaxCompute client, two backslashes (\) are needed to be used as the escape character.
regexp_extract('foothebar', 'foothebar')
-- An error is returned because no part is specified in the pattern.
```

1.6.7.2.12. REGEXP_INSTR

Function declaration:

```
bigint regexp_instr(string source, string pattern[,bigint start_position[, bigint nth_occurrence[, bigint return_option]])
```

Purpose: It is used to return the start or end position of the substring that matches the pattern in the source string from start_position for the nth_occurrence time.

Description:

- source: string type. It indicates a string to be searched.
- pattern: a constant of the string type. If pattern is null, an error is returned.
- start_position: a constant of 'bigint' type. It is the start position for the search. When it is not specified, it is 1 by default. If it is of another type or less than or equal to 0, an error is returned.
- nth_occurrence: a constant of the bigint type. When it is not specified, it is 1 by default, indicating the position where a substring matches pattern in search for the first time. If it is of another type or if it is less than or equal to 0, an error is returned.
- return_option: a constant of the bigint type. The value is either 0 or 1. If it is of another type or the value is not supported, an error is returned. 0 indicates that the start position of the matched substring is returned, and 1 indicates that the end position of the matched substring is returned.

Returned value: bigint type. It is the start or end position of the matched substring in source string according to the type specified by return_option. If any input is NULL, NULL is returned.

Example:

```
regexp_instr("i love www.taobao.com", "o[[:alpha:]]{1}", 3, 2) = 14
```

1.6.7.2.13. REGEXP_SUBSTR

Function declaration:

```
string regexp_substr(string source, string pattern[, bigint start_position[, bigint nth_occurrence]])
```

Purpose: It is used to return the string that matches pattern in the source string from position start_position for the nth_occurence time.

Description:

- source: string type. It indicates a string to be searched.
- pattern: a constant of the string type. It indicates a pattern to be matched. If pattern is null, an error is returned.
- start_position: a constant of the bigint type. It must be greater than 0. If it is another type or if it is less than or equal to 0, an error is reported. When it is not specified, it is regarded as 1 by default, so the matching starts from the first character of 'source'.
- nth_occurrence: a constant of the bigint type. It must be greater than 0. If it is another type or is less than or equal to 0, an error is returned. If it is not specified, it is regarded as 1 by default, indicating that the string in the first match is returned.

Returned value: string type. If any input is NULL, NULL is returned. If there is no matching, NULL is returned.

Example:

```
regexp_substr ("I love aliyun very much", "a[[:alpha:]]{5}") = "aliyun"
regexp_substr('I have 2 apples and 100 bucks!', '[[:blank:]][[:alnum:]]*', 1, 1) = " have"
regexp_substr('I have 2 apples and 100 bucks!', '[[:blank:]][[:alnum:]]*', 1, 2) = "2"
```

1.6.7.2.14. REGEXP_COUNT

Command syntax:

```
bigint regexp_count(string source, string pattern[, bigint start_position])
```

Purpose: It is used to return the number of occurrences that a string pattern appears in the source string, starting from start_position.

Description:

- source: string type. It indicates a string to be searched. For all other input types, an error is returned.
- pattern: string type. It indicates a pattern to be matched. If the pattern is NULL or of another type, an error is returned.
- start_position: bigint start_position must be a number that is greater than 0. Otherwise, an error is returned. If start_position is not specified, the default value is 1 which means starting from the first character of the source string.

Returned value: bigint type. If any input is NULL, NULL is returned. If there is no matching, 0 is returned.

Example:

```
regexp_count('abababc', 'a.c') = 1
regexp_count('abcde', '[[:alpha:]]{2}', 3) = 1
```

1.6.7.2.15. SPLIT_PART

Function declaration:

```
string split_part(string str, string delimiter, bigint start[, bigint end])
```

Purpose: It is used to split a string with the specified delimiter, and return the string between the specified start segment and end segment (inclusive).

Description:

- str: string type. It indicates a string to be split. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. If the input is of any other type, an error is returned.
- delimiter: a constant of the string type. It indicates the delimiter used to split a string. It can be a character or a string. If it is neither a character nor a string, an error is returned.
- start: a constant of the bigint type. It must be greater than 0. If it is not a constant or is of a different type, an error is returned. It indicates the start number (starting from 1) of the segment to be returned. If end is not specified, the segment specified by start is returned.
- end: a constant of the bigint type. It must be greater than or equal to the value of start; otherwise, an error is returned. It indicates the end number of the segment to be returned. If it is not a constant or is of a different type, an error is returned. If end is not specified, the last segment is returned.

Returned value: string type. If any input is NULL, NULL is returned. If delimiter is NULL, the original string is returned.

> **Note**
>
> - If delimiter does not exist in str, and start is set to 1, the entire str is returned. If the input is NULL, NULL is returned.
> - If start is set to a value greater than the number of segments (for example, the string has 6 segments but the start value is greater than 6), NULL is returned.
> - If end is set to a value greater than the number of segments, the string between start and the last segment is returned.

Example:

```
split_part('a,b,c,d', ',', 1) = 'a'
split_part('a,b,c,d', ',', 1, 2) = 'a,b'
split_part('a,b,c,d', ',', 10) = ''
```

1.6.7.2.16. REGEXP_REPLACE

Function declaration:

```
string regexp_replace(string source, string pattern, string replace_string[, bigint occurrence])
```

Purpose: It is used to search a source string for substrings that match a given pattern, replace them with the specified replace_string, and return the result.

Description:

- source: string type. It indicates a string to be replaced.
- pattern: a constant of the string type. It indicates a pattern to be matched. If pattern is null, an error is returned.
- replace_string: string type. It is used to replace the matched pattern.
- occurrence: a constant of the bigint type. It must be greater than or equal to 0. This parameter indicates the number of times at which the substring matches the pattern for replacement with replace_string. If the input value is 0, all matched substrings are replaced. If it is of another type or less than 0, an error is returned. It can be omitted. The default value is 0.

Returned value: string type. When the referenced group does not exist, the replace operation is not performed. When the input parameters source, pattern, and occurrence are NULL, NULL is returned. If replace_string is NULL and the pattern is matched, NULL is returned. If replace_string is NULL but the pattern is not matched, the original string is returned.

> **Note** When the referenced group does not exist, the action is not defined.

Example:

```
regexp_replace("123.456.7890", "([[:digit:]]{3})\\.([[:digit:]]{3})\\.([[:digit:]]{4})", "(\\1)\\2-\\3", 0) = "(123
)456-7890"
regexp_replace("abcd", "(.)", "\\1 ", 0) = "a b c d "
regexp_replace("abcd", "(.)", "\\1 ", 1) = "a bcd"
regexp_replace("abcd", "(.)", "\\2", 1) = "abcd"
-- Only a group is defined in pattern and the referenced second group is not existent.
-- Please avoid this. The result to reference nonexistent group is not defined.
regexp_replace("abcd", "(. *)(.)$", "\\2", 0) = "d"
regexp_replace("abcd", "a", "\\1", 0) = "bcd"
-- No group definition is in pattern, so '\1' references a nonexistent group.
-- Try to avoid this. The result of referencing a nonexistent group is not defined.
```

1.6.7.2.17. SUBSTR

Function declaration:

```
string substr(string str, bigint start_position[, bigint length])
```

Purpose: It is used to return a substring of 'length' from 'str' starting from 'start_position'.

Description:

- str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other types of inputs, an error is returned.

- start_position: bigint type. The start position is 1. If start_position is a negative value, the counting starts from the end to the start of the string and the last character is -1. If the input is of another type, an error is returned.
- length: bigint type. It indicates the length of the substring, which is greater than 0. If it is of another type or less than or equal to 0, an error is returned.

Returned value: string type. If any input is NULL, NULL is returned.

> **Note** If the length is omitted, the substring from start to end is returned.

Example:

```
substr("abc", 2) = "bc"
substr("abc", 2, 1) = "b"
substr("abc",-2,2) = "bc"
substr("abc",-3) = "abc"
```

1.6.7.2.18. TOLOWER

Function declaration:

```
string tolower(string source)
```

Purpose: It is used to convert 'source' into a lowercase string and return the value.

Description:

source: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
tolower("aBcd") = "abcd"
tolower("Haha Cd") = "haha cd"
```

1.6.7.2.19. TOUPPER

Function declaration:

```
string toupper(string source)
```

Purpose: It is used to convert 'source' into an uppercase string and return the value.

Description:

source: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other types of inputs, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
toupper("aBcd") = "ABCD"
toupper("HahaCd") = "HAHACD"
```

1.6.7.2.20. TO_CHAR

Function declaration:

```
string to_char(boolean value)
string to_char(bigint value)
string to_char(double value)
string to_char(decimal value)
```

Purpose: It is used to convert the input of the boolean, bigint, decimal, or double type into a value of the string type.

Description:

value: boolean, bigint, or double type. For all other types of inputs, an error is returned. For more information about the formatted output of data of the datetime type, see Date processing functions — TO_CHAR.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
to_char(123) = '123'
to_char(true) = 'TRUE'
to_char(1.23) = '1.23'
to_char(null) = 'null'
```

1.6.7.2.21. TRIM

Function declaration:

```
string trim(string str)
```

Purpose: It is used to remove the spaces from both ends of 'str'.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other types of inputs, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

1.6.7.2.22. LTRIM

Function declaration:

```
string ltrim(string str)
```

Purpose: It is used to remove the left spaces for input string str.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
select ltrim(' abc ') from dual;
-- Returned result:
+-----+
| _c0 |
+-----+
| abc  |
+-----+
```

1.6.7.2.23. RTRIM

Function declaration:

```
string rtrim(string str)
```

Purpose: It is used to remove the rightmost spaces from the input string 'str'.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
select rtrim('a abc ') from dual;
-- Returned result:
+-----+
| _c0 |
+-----+
| a abc |
+-----+
```

1.6.7.2.24. REVERSE

Function declaration:

```
STRING REVERSE(string str)
```

Purpose: It is used to return a reverse string.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
select reverse('abcedfg') from dual;
-- Returned result:
+-----+
| _c0 |
+-----+
| gfdecba |
+-----+
```

1.6.7.2.25. SPACE

Function declaration:

```
STRING SPACE(bigint n)
```

Purpose: It is used to return a string with 'n' consecutive space characters.

Description:

n: bigint type. The length cannot exceed 2 MB. If the input is NULL, an error is returned.

Returned value: string type.

Example:

```
select length(space(10)) from dual;
-- 10 is returned.
select space(400000000000) from dual;
-- An error is returned as the length exceeds 2 MB.
```

1.6.7.2.26. REPEAT

Function declaration:

```
STRING REPEAT(string str, bigint n)
```

Purpose: It is used to return string 'str' that has been repeated n times.

Description:

- str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.
- n: bigint type. The length cannot exceed 2 MB. If it is NULL, an error is returned.

Returned value: string type.

Example:

```
select repeat('abc',5) from lxw_dual;
-- abcabcabcabcabc is returned.
```

1.6.7.2.27. ASCII

Function declaration:

```
Bigint ASCII(string str)
```

Purpose: It is used to return the ASCII code of the first character of string 'str'.

Description:

str: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

Returned value: bigint type.

Example:

```
select ascii('abcde') from dual;
-- 97 is returned.
```

1.6.7.2.28. URL_ENCODE

Function declaration:

```
STRING URL_ENCODE(STRING input[, STRING encoding])
```

Purpose: It is used to encode the input string in the application/x-www-form-urlencoded MIME format:

- a-z and A-Z remain unchanged.
- ".", "-", "*", and "_" remain unchanged.
- Spaces are converted into "+".
- The rest of the characters are converted into byte values according to the specified encoding. If encoding is not specified, UTF-8 is used by default. In this case, each byte value is represented in the %xy format, where xy represents the hexadecimal form of the character.

Description:

- input: string type.
- encoding: specifies an encoding format. If it is not specified, UTF-8 is used by default.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
url_encode('Example  for url_encode:// (fdsf)') = "%E7%A4%BA%E4%BE%8Bfor+url_encode%3A%2F%2F+
%28fdsf%29"
url_encode('Example for url_encode :// dsf(fasfs)', 'GBK') = "Example+for+url_encode+%3A%2F%2F+dsf
%28fasfs%29"
```

1.6.7.2.29. URL_DECODE

Function declaration:

```
STRING URL_DECODE(STRING input[, STRING encoding])
```

Purpose: It is used to convert an input string from the application/x-www-form-urlencoded MIME format into a normal string. This is the inverse function of URL_ENCODE:

- a-z and A-Z remain unchanged.
- ".", "-", "*", and "_" remain unchanged.
- "+" is converted into a space.
- The %xy formatted sequence is converted into byte values. Consecutive byte values are interpreted as the corresponding strings based on the input encoding.
- Other characters remain unchanged.
- The final returned value of the function is a UTF-8 string.

Description:

- input: string type.

- encoding: specifies an encoding format. If it is not specified, UTF-8 is used by default.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
url_decode('%E7%A4%BA%E4%BE%8Bfor+url_encode%3A%2F%2F+%28fdsf%29')= "Example for url_enco
de:// (fdsf)"
url_decode('Exaple+for+url_encode+%3A%2F%2F+dsf%28fasfs%29', 'GBK') = "Exaple for url_encode ://
dsf(fasfs)" ```
```

1.6.7.2.30. Additional string processing functions

MaxCompute 2.0 provides additional string processing functions. You must add the following SET statement before SQL statements contained in the LPAD, RPAD, and TRANSLATE functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements of the new functions simultaneously.

The string processing functions described in subsequent topics are new in MaxCompute 2.0.

1.6.7.2.31. CONCAT_WS

Command syntax:

```
string concat_ws(string SEP, string a, string b...)
```

Purpose: It is used to join input strings starting from the second with the first string as the separator.

Description:

- SEP: delimiter of the tring type. If it is not specified, an error is returned.
- a, b...: string type. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type. For all other input types, an error is returned.

Returned value: string type. If there is no input or if any input is NULL, NULL is returned.

Example:

```
concat_ws(':','name','bob') = 'name:bob'
concat_ws(':','avg',null,'34')= 'null'
```

1.6.7.2.32. LPAD

Function declaration:

```
string lpad(string a, int len, string b)
```

Purpose: It is used to pad the left side of string a with string b until the new padded string has len bits.

Description:

- len: int type.
- a, b: string type.

Returned value: string type. If len is less than the number of bits in a, a is truncated from the left to obtain a string with the number of bits specified by len. If len is 0, NULL is returned.

Example:

```
lpad('abcdefgh',10,'12')='12abcdefgh'
lpad('abcdefgh',5,'12')='abcde'
lpad('abcdefgh',0,'12')
-- NULL is returned.
```

1.6.7.2.33. RPAD

Function declaration:

```
string rpad(string a, int len, string b)
```

Purpose: It is used to pad the right side of string 'a' with string 'b' until the new padded string has 'len' places.

Description:

- len: int type.
- a, b: string type.

Returned value: string type. If len is smaller than the number of characters in a, a is truncated from the left to obtain a string with the number of characters specified by len. If len is 0, NULL is returned.

Example:

```
rpad('abcdefgh',10,'12')='abcdefgh12'
rpad('abcdefgh',5,'12')='abcde'
rpad('abcdefgh',0,'12')
-- NULL is returned.
```

1.6.7.2.34. REPLACE

Function declaration:

```
string replace(string a, string OLD, string NEW)
```

Purpose: It is used to replace the part of string a that is exactly the same as string OLD with string NEW, and return string a.

Description:

All parameters are of the string type.

Returned value: string type. If any input is NULL, NULL is returned.

Example:

```
replace('abababab','abab','12') = '12ab'
replace('abababab','cdf','123') = 'abababab'
replace('123abab456ab',null,'abab') = 'null'
```

1.6.7.2.35. SOUNDEX

Function declaration:

```
string soundex(string a)
```

Purpose: It is used to convert an ordinary string into a soundex string.

Description:

All parameters are of the string type.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
soundex('hello') = 'H400'
```

1.6.7.2.36. SUBSTRING_INDEX

Function declaration:

```
string substring_index(string a, string SEP, int count))
```

Purpose: It is used to return the substring in 'a' that comes before the 'count' (nth) delimiter ('SEP'). If 'count' is a positive value, it starts from the left of the string. If 'count' is a negative value, it starts from the right of the string.

Description:

- a, SEP: string type.
- count: int type.

Returned value: string type. If the input is NULL, NULL is returned.

Example:

```
substring_index('https://help.aliyun.com', '.', 2) = 'https://help.aliyun'
substring_index('https://help.aliyun.com', '.', -2) = 'aliyun.com'
substring_index('https://help.aliyun.com', null, 2) = 'null'
```

1.6.7.2.37. TRANSLATE

Function declaration:

```
string translate(string|varchar str1, string|varchar str2, string|varchar str3)
```

Purpose: It is used to replace str2 in str1 with str3.

Returned value: STRING type. If any input is NULL, NULL is returned.

Example:

```
translate('MaxComputer','puter','pute')='MaxCompute'
translate('aaa','b','c')='aaa'
translate('MaxComputer','puter',null)=null
```

1.6.7.2.38. JSON_TUPLE

Function declaration:

```
string json_tuple(string json,string key1,string key2,...)
```

Description: This function extracts specific strings from a standard JSON string based on a set of input keys, such as key1 and key2.

Parameters:

- json: a value of the STRING type, which indicates a standard JSON string.
- key: a value of the STRING type, which is used to describe the JSON path. You can enter multiple keys at a time. A key cannot start with a dollar sign ($).

Return value: A value of the STRING type is returned.

Note

- If the json parameter is empty or invalid, NULL is returned.
- If the key parameter is empty or invalid, NULL is returned. If the key parameter does not exist in the JSON string, it is considered invalid.
- If the json parameter is valid and the key parameter exists, the required string is returned.
- This function parses a JSON string the same way as the GET_JSON_OBJECT function for which `set odps.sql.udf.getjsonobj.new=true;` is added. To parse a JSON string multiple times, you must call the GET_JSON_OBJECT function multiple times. However, the JSON_TUPLE function allows you to enter multiple keys at a time and parse the JSON string only once. This improves parsing efficiency.
- JSON_TUPLE is a user-defined table-valued function (UDTF). To select other columns, use JSON_TUPLE with LATERAL VIEW.

Example:

The school table contains the following data:

```
Table: school
+------------+------------+
| Id       | json     |
+------------+------------+
| 1        | {
            "School name": "湖畔大学",
            "Location":"杭州",
            "SchoolRank": "00",
            "Class1":{
              "Student":[{
                "studentId":1,
                "scoreRankIn3Year":[1,2,[3,2,6]]
              }, {
                "studentId":2,
                "scoreRankIn3Year":[2,3,[4,3,1]]
             }]}
           }        |
+------------+------------+
```

Extract JSON objects.

```
SELECT json_tuple(school.json,"SchoolRank","Class1") AS (item0,item1) FROM school;
-- Equivalent to the following statement: SELECT get_json_object(school.json,"$.SchoolRank") item0,ge
t_json_object(school.json,"$.Class1") item1 FROM school;
-- The following result is returned:
+-------+-------+
| item0 | item1 |
+-------+-------+
| 00    | {"Student":[{"studentId":1,"scoreRankIn3Year":[1,2,[3,2,6]]},{"studentId":2,"scoreRankIn3Year":[2
,3,[4,3,1]]}]} |
+-------+-------+
```

Parse JSON data that contains Chinese characters.

```
SELECT json_tuple(school.json,"School name","Location") AS (item0,item1) FROM school;
-- The following result is returned:
+-------+-------+
| item0 | item1 |
+-------+-------+
| 湖畔大学 | 杭州 |
+-------+-------+
```

Parse nested JSON data.

```
SELECT sc.Id, q.item0, q.item1
FROM school sc LATERAL VIEW json_tuple(sc.json,"Class1.Student.[ *].studentId","Class1.Student.[0] .sc
oreRankIn3Year") q AS item0,item1;
-- The following result is returned:
+------------+-------+-------+
| id         | item0 | item1 |
+------------+-------+-------+
| 1          | [1,2] | [1,2,[3,2,6]] |
+------------+-------+-------+
```

Parse JSON data that contains nested arrays.

```
SELECT sc.Id, q.item0, q.item1
FROM school sc LATERAL VIEW json_tuple(sc.json,"Class1.Student[0].scoreRankIn3Year[2]","Class1.Stud
ent[0].scoreRankIn3Year[2][1]") q AS item0,item1;
-- The following result is returned:
+------------+-------+-------+
| id         | item0 | item1 |
+------------+-------+-------+
| 1          | [3,2,6] | 2     |
+------------+-------+-------+
```

1.6.7.3. Date processing functions

1.6.7.3.1. DATEADD

Function declaration:

```
datetime dateadd(datetime date, bigint delta, string datepart)
```

Purpose: It is used to modify date based on delta and datepart.

Description:

- date: This value must be a string type date.If the input is of the string type, it is implicitly converted into a value of the datetime type before this computation. For all other types of inputs, an error is returned.
- delta: bigint type. It indicates the scope of modification. If the input is of the string or double type, it is implicitly converted into a value of the bigint type before this computation. If the input is of another type, an error is returned. If delta is greater than 0, the delta is added to the value. If delta is less than 0, the delta is subtracted from the value.
- datepart: a constant of the string type. This field is set based on the string-datetime conversion convention. yyyy indicates year and mm indicates month. For rules of type conversion, see Conversion between the string type and datetime type. In addition, the extended date format is also supported: year, month or mon, day, and hour. If the parameter value is not a constant or of an unsupported format or another type, an error is returned.

Returned value: datetime type. If any input is NULL, NULL is returned.

Note

- When delta is added to or subtracted from the value, carrying and borrowing are base-10 for year, base-12 for month, base-24 for hour, and base-60 for minute and second. If delta is measured in months, the following calculation is applied: If the month in the datetime value does not cause the day value to become invalid after delta is added, the day value is kept. Otherwise, the day value is adjusted to the last day of the resulting month.
- This field is set based on the string-datetime conversion convention. yyyy indicates the year and mm indicates the month. Unless otherwise specified, all built-in functions related to the datetime type follow this convention. Unless otherwise specified, the datepart of all built-in functions related to the datetime type also supports the extended date format: year, month or mon, day, and hour.

Example:

```
If trans_date = 2017-02-28 00:00:00:
dateadd(trans_date, 1, 'dd') = 2017-03-01 00:00:00
-- Add one day. The result is beyond the last day of February. The actual value is the first day of next month.
dateadd(trans_date, -1, 'dd') = 2017-02-27 00:00:00
-- Subtract one day.
dateadd(trans_date, 20, 'mm') = 2018-10-28 00:00:00
-- 20 months are added. The month overflows, and 1 is added to the year.
trans_date = 2017-02-28 00:00:00, dateadd(transdate, 1, 'mm') = 2017-03-28 00:00:00
trans_date = 2017-01-29 00:00:00, dateadd(transdate, 1, 'mm') = 2017-02-28 00:00:00
-- February has 28 days only, so the last day of the month is returned.
trans_date = 2017-03-30 00:00:00, dateadd(transdate, -1, 'mm') = 2017-02-28 00:00:00
```

The values of trans_date used only serve as examples. The datetime examples in this document use simple formats. In MaxCompute SQL, a constant cannot be of the datetime type. The following syntax is incorrect:

```
select dateadd(2017-03-30 00:00:00, -1, 'mm') from tbl1;
```

If you must use a constant of the datetime type, use the following method:

```
select dateadd(cast("2017-03-30 00:00:00" as datetime), -1, 'mm') from tbl1;
-- The String type constant is converted to datatime type by explicit conversion.
```

1.6.7.3.2. DATEDIFF

Function declaration:

```
bigint datediff(datetime date1, datetime date2, string datepart)
```

Purpose: It is used to calculate the difference between date1 and date2 based on the specified datepart.

Description:

- date1 and date2: minuend and subtrahend of the datetime type respectively. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other input types, an error is returned.
- datapart: A constant of the string type. It supports the extended date format. If datepart is not in the specified format or is of another type, an error is returned.

Returned value: bigint type. If any input is NULL, NULL is returned. If date1 is less than date2, the returned value may be negative.

> **Note** The lower unit part is truncated based on 'datepart' in the computation process and then the result is calcualted.

Example:

```
If start = 2017-12-31 23:59:59 and end = 2018-01-01 00:00:00:
datediff(end, start, 'dd') = 1
datediff(end, start, 'mm') = 1
datediff(end, start, 'yyyy') = 1
datediff(end, start, 'hh') = 1
datediff(end, start, 'mi') = 1
datediff(end, start, 'ss') = 1
datediff('2017-05-31 13:00:00', '2017-05-31 12:30:00', 'ss') = 1800
datediff('2017-05-31 13:00:00', '2017-05-31 12:30:00', 'mi') = 30
```

1.6.7.3.3. DATEPART

Function declaration:

```
bigint datepart(datetime date, string datepart)
```

Purpose: It is used to extract the value of the specified datepart in date.

Description:

- date: datetime type. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other input types, an error is returned.
- datepart: a constant of the string type. It supports the extended date format. If datepart is not in the specified format or is of another type, an error is returned.

Returned value: bigint type. If any input is NULL, NULL is returned.

Example:

```
datepart('2017-06-08 01:10:00', 'yyyy') = 2017
datepart('2017-06-08 01:10:00', 'mm') = 6
```

1.6.7.3.4. DATETRUNC

Function declaration:

```
datetime datetrunc (datetime date,string datepart)
```

Purpose: It is used to return the value of a date after the specified datepart is truncated.

Description:

- date: datetime type. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other input types, an error is returned.
- datepart: a constant of the string type. It supports the extended date format. If datepart is not in the specified format or is of another type, an error is returned.

Returned value: datetime type. If any input is NULL, NULL is returned.

Example:

```
datetrunc('2017-12-07 16:28:46', 'yyyy') = 2017-01-01 00:00:00
datetrunc('2017-12-07 16:28:46', 'month') = 2017-12-01 00:00:00
datetrunc('2017-12-07 16:28:46', 'DD') = 2017-12-07 00:00:00
```

1.6.7.3.5. GETDATE

Function declaration:

```
datetime getdate()
```

Purpose: It is used to obtain the current system time. Use UTC+8 as the standard time of MaxCompute.

Returned value: the current date and time of the datetime type.

> **Note** In a MaxCompute SQL task (executed in a distributed manner), 'getdate' always returns a fixed value. The returned result is any time in MaxCompute. The time returned is precise to the second. In later versions, the time will be precise to the milisecond.

1.6.7.3.6. ISDATE

Function declaration:

```
boolean isdate(string date, string format)
```

Purpose: It is used to determine whether a date string can be converted into a date value based on the corresponding format string. If the conversion can be performed, true is returned. Otherwise, false is returned.

Description:

- date: This value must be a string type date. If the input is of the bigint, decimal, double, or

datetime type, it is implicitly converted into a value of the string type before this computation. For all other input types, an error is returned.

- format: a constant of the string type. The extended date format is not supported. If it is of another type or an unsupported format, an error is returned. If there are redundant format strings appearing in 'format', the date value corresponding to the first format string is used. Other strings are taken as delimiters. If isdate("1234-yyyy", "yyyy-yyyy"), true is returned.

Returned value: boolean type. If any input is NULL, NULL is returned.

1.6.7.3.7. LASTDAY

Function declaration:

```
datetime lastday(datetime date)
```

Purpose: It is used to return the last day of the current month to which the date belongs. The value is accurate to day. The hour, minute, and second part is expressed as 00:00:00.

Description:

date: datetime type. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other input types, an error is returned.

Returned value: datetime type. If the input is NULL, NULL is returned.

1.6.7.3.8. TO_DATE

Function declaration:

```
datetime to_date(string date, string format)
```

Purpose: It is used to convert the 'date' string into a date value.

Description:

- date: string type. It indicates the date value of the string type to be converted. If the input is of the bigint, decimal, double, or datetime type, it is implicitly converted into a value of the string type before this computation. For all other types of inputs or NULL, an error is returned.
- format: a constant of the string type in the date format. For all other types of inputs and non-constant values, an error is returned. It does not support the extended date format. Other characters are ignored as invalid characters in parsing. The format parameter must contain yyyy. Otherwise, an error is returned. If there are redundant format strings in the format, the corresponding date value of the first format string is used, and the rest are processed as separators. For example, to_date('1234-2234', 'yyyy-yyyy') returns '1234-01-01 00:00:00'.

Returned value: datetime type. The format is *yyyy-mm-dd hh:mi:ss*. If any input is NULL, NULL is returned.

Example:

```
to_date('Alibaba2017-12*03', 'Alibabayyyy-mm*dd') = 2017-12-03 00:00:00
to_date('20170718', 'yyyymmdd') = 2017-07-18 00:00:00
to_date('201707182030','yyyymmddhhmi')=2017-07-18 20:30:00
to_date('2017718', 'yyyymmdd')
-- Invalid format. NULL is returned.
to_date('Alibaba2017-12*3', 'Alibabayyyy-mm*dd')
-- Invalid format. NULL is returned.
to_date('2017-24-01', 'yyyy')
-- Invalid format. NULL is returned.
```

1.6.7.3.9. TO_CHAR

Function declaration:

```
string to_char(datetime date, string format)
```

Purpose: It is used to convert a value of the date type into a string based on the specified format.

Description:

- date: date value of the datetime type to be converted. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other types of inputs, an error is returned.
- format: a constant of the string type. If it is not a constant or is of a different type, an error is returned. In format, the date format part is replaced with the corresponding data and other characters are output directly.

Returned value: string type. If any input is NULL, NULL is returned.

Example:

```
to_char('2017-12-03 00:00:00', 'Alibabayyyy-mm*dd') = 'Alibaba2017-12*03'
to_char('2017-07-18 00:00:00', 'yyyymmdd') = '20170718'
to_char('Alibaba 2017-12*3', 'Alibaba yyyy-mm*dd')
-- Null is returned.
to_char('2017-24-01', 'yyyy')
-- Null is returned.
to_char('2017718', 'yyyymmdd')
-- Null is returned.
```

> **Note** For more information about conversion from other types into the string type, see String functions — TO_CHAR.

1.6.7.3.10. UNIX_TIMESTAMP

Function declaration:

```
bigint unix_timestamp(datetime date)
```

Purpose: It is used to convert a date into a datetime value of the integer type in the Unix format.

Description:

date: datetime type. It indicates the date. If the input is a string, it is implicitly converted into a value of the datetime type before this computation. For all other input types, an error is returned.

Returned value: bigint type. It indicates the date value in Unix format. If date is NULL, NULL is returned.

1.6.7.3.11. FROM_UNIXTIME

Function declaration:

```
datetime from_unixtime(bigint unixtime)
```

Purpose: It is used to convert a Unix date value from the BIGINT type to the DATETIME type.

Description:

unixtime: BIGINT type. It is a date value in the Unix format. If the input is of the STRING, DECIMAL, or DOUBLE type, it is implicitly converted into a value of the BIGINT type before computation.

Returned value: DATETIME type. If unixtime is NULL, NULL is returned.

> **Note** In the HIVE-compatible mode (where `set odps.sql.hive.compatible=true;` has been executed), if the input is of the STRING type, a date value of the STRING type is returned.

Example:

```
from_unixtime(123456789) = 1973-11-30 05:33:09;
```

1.6.7.3.12. WEEKDAY

Function declaration:

```
bigint weekday (datetime date)
```

Purpose: It is used to return the day of week for the specified date.

Description:

date: datetime type. If the input is of the string type, it is implicitly converted to a value of the datetime type before this computation. For all other input types, an error is returned.

Returned value: bigint type. If the input is NULL, NULL is returned. Monday is the first day of a week and the returned value is 0. Days are numbered in ascending order starting from 0. If the day is Sunday, the returned value is 6.

1.6.7.3.13. WEEKOFYEAR

Function declaration:

```
bigint weekofyear(datetime date)
```

Purpose: It is used to return the calendar week of the year that the specified date falls in. The system uses Monday as the first day of the week.

> **Note** If a week extends into the next year, the week belongs to the year containing four days or more. If more days fall in the first year, the week is considered as the last week of the first year. If more days fall in the second year, the week is considered as the first week of the second year.

Description:

date: the date of the datetime type. If the input is of the string type, it is implicitly converted to a value of the datetime type before this computation. For all other input types, an error is returned.

Returned value: bigint type. If the input is NULL, NULL is returned.

Example:

```
select weekofyear(to_date("20171229", "yyyymmdd")) from dual;
Returned value:
+------------+
| _c0        |
+------------+
| 1          |
+------------+
-- 20171229 is in year 2017, but the most days of the week are in year 2018. Therefore, the returned val
ue is 1, which indicates the first week of year 2018.
select weekofyear(to_date("20171231", "yyyymmdd")) from dual;
-- 1 is returned.
select weekofyear(to_date("20181229", "yyyymmdd")) from dual;
-- The returned value is 53.
```

1.6.7.3.14. Additional date functions

MaxCompute 2.0 provides additional date functions. You must add the following SET statement before SQL statements contained in the date functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements of the new functions simultaneously.

Example:

```
set odps.sql.type.system.odps2=true;
select year('2017-01-01 12:30:00') = 2017 from dual;
```

The date functions described in subsequent topics are new in MaxCompute 2.0.

1.6.7.3.15. YEAR

Function declaration:

```
INT year(string date)
```

Purpose: It is used to return the year of the specified date.

Description:

date: the date of the string type. The date format must include yyyy-mm-dd and have no redundant strings. Otherwise, NULL is returned.

Returned value: int type.

Example:

```
year('2017-01-01 12:30:00') = 2017
year('2017-01-01') = 2017
year('17-01-01') = 17
year(2017-01-01) = null
year('2017/03/09') = null
year(null) = null
```

1.6.7.3.16. QUARTER

Command syntax:

```
int quarter(datetime/timestamp/string date)
```

Purpose: It is used to return the quarter of the input date, ranging from 1 to 4.

Description:

date: datetime, timestamp, or string type. The date format must include yyyy-mm-dd and have no redundant strings. Otherwise, NULL is returned.

Returned value: int type. If the input is NULL, NULL is returned.

Example:

```
quarter('2017-11-12 10:00:00') = 4
quarter('2017-11-12') = 4
```

1.6.7.3.17. MONTH

Function declaration:

```
INT month(string date)
```

Purpose: It is used to return the month of the input date.

Description:

date: This value must be a date of the string type. For all other input types, an error is returned.

Returned value: int type.

Example:

```
month('2017-09-01') = 9
month('20170901') = null
```

1.6.7.3.18. DAY

Function declaration:

```
INT day(string date)
```

Purpose: It is used to return the day of the input date.

Description:

date: This value must be a string type date. For all other input types, an error is returned.

Returned value: int type.

Example:

```
day('2017-09-01') = 1
day('20170901') = null
```

1.6.7.3.19. DAYOFMONTH

Function declaration:

```
INT dayofmonth(date)
```

Purpose: It is used to return the day of the month for the input date.

Description:

date: This value must be a string type date. For all other input types, an error is returned.

Returned value: int type.

Example:

```
dayofmonth('2017-09-01') = 1
dayofmonth('20170901') = null
```

1.6.7.3.20. HOUR

Function declaration:

```
INT hour(string date)
```

Purpose: It is used to return the hour of the input date.

Description:

date: This value must be a string type date. For all other input types, an error is returned.

Returned value: int type.

Example:

```
hour('2017-09-01 12:00:00') = 12
hour('12:00:00') = 12
hour('20170901120000') = null
```

1.6.7.3.21. MINUTE

Function declaration:

```
INT minute(string date)
```

Purpose: It is used to return the minute of the input date.

Description:

date: This value must be a string type date. For all other input types, an error is returned.

Returned value: int type.

Example:

```
minute('2017-09-01 12:30:00') = 30
minute('12:30:00') = 30
minute('20170901120000') = null
```

1.6.7.3.22. SECOND

Function declaration:

```
INT second(string date)
```

Purpose: It is used to return the second of the input date.

Description:

date: This value must be a string type date. For all other input types, an error is returned.

Returned value: int type.

Example:

```
second('2017-09-01 12:30:45') = 45
second('12:30:45') = 45
second('20170901123045') = null
```

1.6.7.3.23. FROM_UTC_TIMESTAMP

Function declaration:

```
timestamp from_utc_timestamp({any primitive type}*, string timezone)
```

Purpose: It is used to convert a UTC timestamp to a timestamp for a specified timezone.

Description:

- {any primitive type}*: the timestamp. The type can be TIMESTAMP, DATETIME, TINYINT, SMALLINT, INT, or BIGIN.
- timezone: Specifies the destination timezone, such as PST.

Returned value: DATETIME type.

Example:

```
select from_utc_timestamp(1501557840,'PST') = '1970-01-18 09:05:57.84'
select from_utc_timestamp('1970-01-30 16:00:00','PST') = '1970-01-30 08:00:00.0'
select from_utc_timestamp('1970-01-30','PST') = '1970-01-29 16:00:00.0'
```

1.6.7.3.24. CURRENT_TIMESTAMP

Function declaration:

```
timestamp current_timestamp()
```

Purpose: The current timestamp is returned as a Timestamp-type value. The value is not fixed.

Returned value: timestamp type.

Example:

```
select current_timestamp() from dual;
-- '2017-08-03 11:50:30.661'is returned.
```

1.6.7.3.25. ADD_MONTHS

Function declaration:

```
string add_months(string startdate, int nummonths)
```

Purpose: It is used to return the date, which is 'nummonths' months later than 'startdate'.

Description:

- startdate: This value must be a string type date. The date format must contain yyyy-mm-dd. Otherwise, NULL is returned.
- num_months: int type.

Returned value: This value must be a string type date. The format is yyyy-mm-dd.

Example:

```
Add_months ('2017-02-14', 3) = '2017-05-14'
add_months('17-2-14',3) = '0017-05-14'
add_months('2017-02-14 21:30:00',3) = '2017-05-14'
add_months('20170214',3) = null
```

1.6.7.3.26. LAST_DAY

Function declaration:

```
string last_day(string date)
```

Purpose: It is used to return the last date of the month.

Description:

date: string type. The format is yyyy-MM-dd HH:mi:ss or yyyy-mm-dd.

Returned value: This value must be a datetime type date. The format is yyyy-mm-dd.

Example:

```
last_day('2017-03-04') = '2017-03-31'
last_day('2017-07-04 11:40:00') = '2017-07-31'
last_day('20170304') = null
```

1.6.7.3.27. NEXT_DAY

Function declaration:

```
string next_day(string startdate, string week)
```

Purpose: It is used to return the next date that is later than startdate and matches the week value. That is, the date of the day specified of the next week.

Description:

- startdate: string type. The format is yyyy-MM-dd HH:mi:ss or yyyy-mm-dd.
- week: string type. The name of a day, or the first 2 or 3 letters of the day, for example, Mo, TUE, or FRIDAY.

Returned value: This value must be a string type date. The format is yyyy-mm-dd.

Example:

```
next_day('2017-08-01','TU') = '2017-08-08'
next_day('2017-08-01 23:34:00','TU') = '2017-08-08'
Next_day ('20170801 ', 'tu') = NULL
```

1.6.7.3.28. MONTHS_BETWEEN

Function declaration:

```
double months_between(datetime/timestamp/string date1, datetime/timestamp/string date2)
```

Purpose: It is used to return the number of months between date1 and date2.

Description:

- date1: datetime, timestamp, or string type. The format is yyyy-MM-dd HH:mi:ss or yyyy-mm-dd.
- date2: datetime, timestamp, or string type. The format is yyyy-MM-dd HH:mi:ss or yyyy-mm-dd.

Returned value: double type.

- If 'date1' is later than 'date2', the returned value is positive. If 'date2' is later than 'date1', the returned value is negative.
- When date1 and date2 correspond to the last days of two months, the returned value is an integer representing the number of months. Otherwise, the formula is (date1 - date2)/31.

Example:

```
months_between('1997-02-28 10:30:00', '1996-10-30') = 3.9495967741935485
months_between('1996-10-30','1997-02-28 10:30:00' ) = -3.9495967741935485
months_between('1996-09-30','1996-12-31') = -3.0
```

1.6.7.3.29. EXTRACT

Function declaration:

```
INT EXTRACT(<datepart> from <timestamp>)
```

Description: This function extracts the part specified by datepart from the time specified by timestamp.

Parameters:

- datepart: a value that can be set to a time unit, such as YEAR, MONTH, DAY, HOUR, or MINUTE
- timestamp: a value of the TIMESTAMP type

Return value: A value of the INT type is returned.

Example:

```
SET odps.sql.type.system.odps2=true;
SELECT  extract(YEAR FROM '2019-05-01 11:21:00') year
       ,extract(MONTH FROM '2019-05-01 11:21:00') month
       ,extract(DAY FROM '2019-05-01 11:21:00') day
       ,extract(HOUR FROM '2019-05-01 11:21:00') hour
       ,extract(MINUTE FROM '2019-05-01 11:21:00') minute;
-- The following result is returned:
+------+-------+------+------+--------+
| year | month | day  | hour | minute |
+------+-------+------+------+--------+
| 2019 | 5     | 1    | 11   | 21     |
+------+-------+------+------+--------+
```

If the time value specified in the SQL statement is invalid or exceeds the specified range, the return value is the remainder obtained by dividing the specified time value by the maximum value in the time range.

Example:

```
SET odps.sql.type.system.odps2=true;
SELECT  extract(HOUR FROM '2019-05-01 31:01:01') hour
       ,extract(MINUTE FROM '2019-05-01 23:61:01') minute;
-- The following result is returned:
+------+-------+
| hour | minute|
+------+-------+
| 7    | 1     |
+------+-------+
-- The maximum value of hour is 24, and the specified time value is 31. The return value is 7 (31/24).
-- The maximum value of minute is 60, and the specified time value is 61. The return value is 1 (61/60).
```

1.6.7.4. Window functions

1.6.7.4.1. Overview

In MaxCompute SQL statements, you can use the window function to analyze and process data flexibly. The window function can only appear in SELECT clauses. It does not support nested Window or aggregation functions. The window function cannot be used with the same-level aggregation functions at the same time.

A MaxCompute SQL statement supports up to five window functions.

Syntax:

```
window_func() over (partition by col1, [col2...]
[order by col1 [asc|desc][, col2[asc|desc]...]] windowing_clause)
```

Description:

- PARTITION BY specifies partition columns. The rows on which the partition column values are the same are considered to be in the same window. A window can contain up to 100 million rows of data (we recommend that the number of rows does not exceed 5 million). Otherwise, an error is returned.
- Use ORDER BY to specify the rule for sorting data in a window.
- You can use ROWS in windowing_clause to specify the partitioning method. There are two methods:
  - rows between x preceding|following and y preceding|following indicates a window range from the xth row preceding or following the current row to the yth row preceding or following the current row.
  - rows x preceding|following indicates a window range from the xth row preceding or following the current row to the current row.

> **Note**
>
> - x and y must be integer constants greater than or equal to 0. Their values range from 0 to 10,000. 0 indicates the current row.
> - You must specify ORDER BY before using ROWS to specify a window range.
> - Not all window functions open windows using the method specified by ROWS. The method is only supported by the following functions: AVG, COUNT, MAX, MIN, STDDEV, and SUM.

1.6.7.4.2. COUNT

Command syntax:

```
bigint count([distinct] expr) over(partition by col1[, col2...]
[order by col1 [asc|desc][, col2[asc|desc]...]] [windowing_clause])
```

Purpose: It is used to return the number of values on the expr column.

Description:

- expr: any type. When it is NULL, this row is not involved in computation. If the distinct keyword is specified, this parameter indicates that only distinct values are counted.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc],col2[asc|desc]: The count value of expr in the current window is returned if ORDER BY is not set. The returned results are sorted in the specified order if ORDER BY is specified, and the value is the count value from the start row to the current row in the current window.

Returned value: bigint.

> **Note** If the distinct keyword is specified, ORDER BY cannot be used.

Example:

The user_id column of the bigint type exists in the test_src table.

```
select user_id,count(user_id) over (partition by user_id) as count from test_src;
+---------+------------+
| user_id |  count     |
+---------+------------+
| 1       | 3          |
| 1       | 3          |
| 1       | 3          |
| 2 | 1 |
| 3 | 1 |
+---------+------------+
+---------+------------+
-- If ORDER BY is not specified, the number of values on the user_id column from the current partition i
s returned.
select user_id,count(user_id) over (partition by user_id order by user_id) as count from test_src;
+---------+------------+
| user_id | count      |
+---------+------------+
| 1 | 1 | -- start row of the window
| 1 | 2 | --two records exist from start row to current row. Return 2.
| 1       | 3          |
| 2       | 1          |
| 3       | 1          |
+---------+------------+
-- If ORDER BY is specified, the count value from the start row to the current row from the current parti
tion is returned.
```

1.6.7.4.3. AVG

Function declaration:

```
avg([distinct] expr) over(partition by col1[, col2...]
[order by col1 [asc|desc] [, col2[asc|desc]...]] [windowing_clause])
```

Purpose: It is used to calculate the average value.

Description:

- distinct: If the distinct keyword is specified, this parameter indicates that the average value of distinct values is calculated.
- expr: double type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before computation. If the input is of another type, an error is returned. If the input is NULL, this row is not used in computation. The input cannot be of the boolean type.
- partition by col1[, col2]...: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The count value of expr in the current window is returned if ORDER BY is not set. The returned results are sorted in the specified order if ORDER BY is specified, and the value is the count value from the start row to the current row in the current window.

Returned value: double type.

> **Note** If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.4. MAX

Function declaration:

```
max([distinct] expr) over(partition by col1[, col2...]
[order by col1 [asc|desc][, col2[asc|desc]...]] [windowing_clause])
```

Purpose: It is used to return the maximum value.

Description:

- expr: any types except the boolean type. If the value is NULL, the corresponding row is not involved in the operation. If the distinct keyword is specified, this parameter indicates that the maximum value of the distinct values is taken (whether this parameter is set or not does not affect the result).
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The maximum value in the current window is returned if ORDER BY is not set. If ORDER BY is set, the returned results are sorted in the specified order, and the values are the maximum values from the start row to the current row in the current window.

Returned value: The type is the same as that of expr.

> Note If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.5. MIN

Function declaration:

```
min([distinct] expr) over(partition by col1[, col2...]
[order by col1 [asc|desc][, col2[asc|desc]...]] [windowing_clause])
```

Purpose: It is used to return the minimum value.

Description:

- expr: any types except the boolean type. If the value is NULL, the corresponding row is not involved in the operation. If the distinct keyword is specified, this parameter indicates that the minimum value of distinct values is taken (whether this parameter is set or not does not affect the result).
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The minimum value in the current window is returned if ORDER BY is not set. If ORDER BY is set, the returned results are sorted in the specified order, and the returned value is the minimum value in the current window from the start row to the current row.

Returned value: The type is the same as that of expr.

> Note If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.6. MEDIAN

Function declaration:

```
double median(double number) over(partition by col1[, col2...])
decimal median(decimal number) over(partition by col1[,col2...])
```

Purpose: It is used to calculate the median.

Description:

- number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned. If the input is NULL, NULL is returned.
- partition by col1[, col2...]: specifies the partitions used in the computation.

Returned value: double type.

1.6.7.4.7. STDDEV

Function declaration:

```
double stddev([distinct] expr) over(partition by col1[, col2...] [order by col1 [asc|desc][, col2[asc|desc]...
]] [windowing_clause])
decimal stddev([distinct] expr) over(partition by col1[,col2...] [order by col1 [asc|desc][, col2[asc|desc]...
]] [windowi ng_clause])
```

Purpose: It is used to calculate the population standard deviation.

Description:

- expr: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned. If the input value is NULL, then NULL is returned. If the distinct keyword is specified, this parameter indicates that the population standard deviation of distinct values is calculated.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The population standard deviation of the current window is returned if ORDER BY is not set. If ORDER BY is set, the returned results are sorted in the specified order, and the values are the population standard deviation of the start row to the current row in the current window.

Returned value: When the input is of the decimal type, a value of the decimal type is returned. Otherwise, a value of the double type is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.8. STDDEV_SAMP

Function declaration:

```
double stddev_samp([distinct] expr) over(partition by col1[, col2...] [order by col1 [asc|desc][, col2[asc|
desc]...]] [windowing_clause])
decimal stddev_samp([distinct] expr) over(partition by col1[,col2...] [order by col1 [asc|desc][, col2[asc|
desc]...]] [windowing_clause])
```

Purpose: It is used to calculate the sample standard deviation.

Description:

- expr: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned. If the input is NULL, NULL is returned. If the distinct keyword is specified, this parameter indicates that the sample standard deviation of distinct values is calculated.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The sample standard deviation of the current window is returned if ORDER BY is not set. If ORDER BY is set, the returned results are sorted in the specified order, and the values are the sample standard deviation of the start row to the current row in the current window.

Returned value: When the input is of the decimal type, a value of the decimal type is returned. Otherwise, a value of the double type is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.9. SUM

Function declaration:

```
sum([distinct] expr) over(partition by col1[, col2...]
[order by col1 [asc|desc][, col2[asc|desc]...]] [windowing_clause])
```

Purpose: It is used calculate the sum.

Description:

- expr: double, decimal, or bigint type. If the input is of the string type, it is implicitly converted into a value of the double type before computation. If the input is of another type, an error is returned. If the value is NULL, this row is not calculated. If the distinct keyword is specified, this parameter indicates that the sum of distinct values is calculated.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: The sum of the expr value in the current window is returned if ORDER BY is not set. If ORDER BY is set, the returned results are sorted in the order specified. The returned results are the cumulative sum of start row to the current row in the current window.

Returned value: When the input is of the bigint type, a value of the bigint type is returned. When the input is of the double or string type, a value of the double type is returned.

> **Note** If the distinct keyword is specified, ORDER BY cannot be set.

1.6.7.4.10. DENSE_RANK

Function declaration:

```
bigint dense_rank() over(partition by col1[, col2...]
order by col1 [asc|desc][, col2[asc|desc]...])
```

Purpose: It is used to calculate the consecutive ranking of values. The data in the same row of col2 has the same rank.

Description:

- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: This parameter specifies the value for deciding the ranking.

Returned value: bigint type.

Example:

The emp table contains the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

To obtain their serial number, the employees must be group by their departments and sorted by SAL in descending order.

```
SELECT deptno
      , ename
      , sal
      , DENSE_RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS nums
-- DEPTNO (department) is the partition used in the computation, and SAL (salary) is used as basis for sorting returned results.
   FROM emp;
-- Returned result:
+------------+-------+------------+------------+
| deptno     | ename | sal        | nums       |
+------------+-------+------------+------------+
| 10         | JACCKA | 5000.0     | 1          |
| 10 | King | 5000.0 | 1 |
| 10 | CLARK | 2450.0 | 2 |
| 10 | WELAN | 2450.0 | 2 |
| 10 | TEBAGE | 1300.0 | 3 |
10 | Miller | 1300.0 | 3 |
| 20         | SCOTT | 3000.0     | 1          |
| 20 | Ford | 3000.0 | 1 |
| 20 | JONES | 2975.0 | 2 |
| 20 | ADAMS | 1100.0 | 3 |
| 20 | SMITH | 800.0 | 4 |
| 30         | BLAKE | 2850.0     | 1          |
| 30         | ALLEN | 1600.0     | 2          |
| 30         | TURNER | 1500.0     | 3          |
| 30         | MARTIN | 1250.0     | 4          |
| 30 | WARD | 1250.0 | 4 |
| 30 | JAMES | 950.0 | 5 |
+------------+-------+------------+------------+
```

1.6.7.4.11. RANK

Command syntax:

```
bigint rank() over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...])
```

Purpose: It is used to return a ranking value. The ranking of the same row data with col2 drops.

Description:

- partition by col2[, col2..]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: specifies the rule for deciding the ranking.

Returned value: bigint type.

Example:

Table emp contains the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Now group the employees by department. Sort the employees in each group in descending order based on the salary. Each employee obtains a number that represents their position in the group.

```
SELECT deptno
      , ename
      , sal
      , RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) AS nums
-- DEPTNO (department) is the partitioning column. The sal column is sorted to generate the ranking v
alue for each employee.
   FROM emp;
-- Returned result:
+------------+-------+------------+------------+
| deptno     | ename | sal        | nums       |
+------------+-------+------------+------------+
| 10         | JACCKA | 5000.0     | 1          |
| 10         | KING  | 5000.0     | 1          |
| 10         | CLARK | 2450.0     | 3          |
| 10 | WELAN | 2450.0 | 3 |
| 10         | TEBAGE | 1300.0     | 5          |
| 10 | MILLER | 1300.0 | 5 |
| 20         | SCOTT | 3000.0     | 1          |
| 20         | FORD  | 3000.0     | 1          |
| 20         | JONES | 2975.0     | 3          |
| 20         | ADAMS | 1100.0     | 4          |
| 20         | SMITH | 800.0      | 5          |
| 30         | BLAKE | 2850.0     | 1          |
| 30         | ALLEN | 1600.0     | 2          |
| 30         | TURNER | 1500.0     | 3          |
| 30         | MARTIN | 1250.0     | 4          |
| 30         | WARD  | 1250.0     | 4          |
| 30         | JAMES | 950.0      | 6          |
+------------+-------+------------+------------+
```

1.6.7.4.12. LAG

Function declaration:

```
lag(expr, bigint offset, default) over(partition by col1[, col2...] [order by col1 [asc|desc][, col2[asc|desc]
...]])
```

Purpose: It is used to retrieve the value in the row with a negative offset from the current row. For example, if the current row is rn, the value retrieved is from the row rn - offset.

Description:

- expr: any type.
- offset: a constant of the bigint type. If the input is of the string or double type, it is implicitly converted into a value of the bigint type before computation, and the offset is greater than 0.
- default: a constant. It specifies the default value when the offset is out of the valid range. The default value is NULL.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: indicates the sorting order of the returned results.

Returned value: The type is the same as that of expr.

1.6.7.4.13. LEAD

Function declaration:

```
lead(expr, bigint offset, default) over(partition by col1[, col2...][order by col1 [asc|desc][, col2[asc|desc]
...]])
```

Purpose: It is used to retrieve the value in the row with a positive offset from the current row. For example, if the current row is rn, the value retrieved is from the row rn + offset.

Description:

- expr: any type.
- offset: a constant of the bigint type. If the input is of the string or double type, it is implicitly converted into a value of the bigint type before computation, and the offset is greater than 0.
- default: a constant. It specifies the default value when the offset is out of the valid range.
- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], > col2[asc|desc]: indicates the sorting order of the returned results.

Returned value: The type is the same as that of expr.

Example:

```
select c_double_a,c_string_b,c_int_a,lead(c_int_a,1) over(partition by c_double_a order by c_string_b) fr
om dual;
select c_string_a,c_time_b,c_double_a,lead(c_double_a,1) over(partition by c_string_a order by c_time_
b) from dual;
select c_string_in_fact_num,c_string_a,c_int_a,lead(c_int_a) over(partition by c_string_in_fact_num ord
er by c_string_a) from dual;
```

1.6.7.4.14. PERCENT_RANK

Function declaration:

```
percent_rank() over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...])
```

Purpose: It is used to return the relative ranking of a row in a group of data.

Description:

- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], col2[asc|desc]: specifies the value for the ranking.

Returned value: double type. Value range: 0 to 1. The relative ranking is calculated using the following formula: (rank-1)/(number of rows -1).

> **Note** The number of rows in a window cannot exceed 10,000,000.

1.6.7.4.15. ROW_NUMBER

Function declaration:

```
row_number() over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...])
```

Purpose: It is used to calculate the row number, which starts from 1.

Description:

- partition by col1[, col2...]: specifies the partitions used in the computation.
- order by col1 [asc|desc], > col2[asc|desc]: indicates the sorting value of the returned result.

Returned value: bigint type.

Example:

If table emp contains the following data:

```
| Empno | ename | job | Mgr | hiredate | Sal | REM | deptno |
7369, Smith, clerk, maid-12-17 00:00:00, 800, 20
7499, Allen, salesman, maid-02-20 00:00:00, 1600,300, 30
7521, Ward, salesman, maid-02-22 00:00:00, 1250,500, 30
7566, Jones, Manager, fig-04-02 00:00:00, 2975, 20
7654 Martin, salesman, fig-09-28 00:00:00, fig, 30
7698, Blake, Manager, fig-05-01 00:00:00, 2850, 30
7782, Clark, Manager, fig-06-09 00:00:00, 2450, 10
7788, Scott, analyst, fig-04-19 00:00:00, 3000, 20
00:00:00, King, President, 1991-11-17 5000, 7839, 10
7844, Turner, salesman, fig-09-08 00:00:00, 1500,0, 30
7876, Adams, clerk, maid-05-23 00:00:00, 1100, 20
7900 James, clerk, maid-12-03 00:00:00, 950, 30
7902 Ford, analyst, fig-12-03 00:00:00, 3000, 20
7934 Miller, clerk, fig-01-23 00:00:00, 1300, 10
7948, jaccka, clerk, fig-04-12 00:00:00, 5000, 10
7956, welan, clerk, fig-07-20 00:00:00, 2450, 10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Now, all employees need to be grouped by department, and each group must be sorted in descending order according to SAL to obtain the serial number in own group.

```
SELECT deptno
      , ename
      ,sal
      , ROW_NUMBER() OVER (PARTITION BY deptno ORDER BY sal DESC) AS nums
-- DEPTNO (department) is the partition used in the computation, and SAL (salary) is used as basis for sorting results.
   FROM emp;
-- Returned result:
+------------+-------+------------+------------+
| deptno    | ename | sal       | nums      |
+------------+-------+------------+------------+
| 10 | JACCKA | 5000.0 | 1 |
| 10 | KING | 5000.0 | 2 |
| 10 | CLARK | 2450.0 | 3 |
| 10 | WELAN | 2450.0 | 4 |
| 10 | TEBAGE | 1300.0 | 5 |
| 10 | MILLER | 1300.0 | 6 |
| 20 | SCOTT | 3000.0 | 1 |
| 20 | FORD | 3000.0 | 2 |
| 20 | JONES | 2975.0 | 3 |
| 20 | ADAMS | 1100.0 | 4 |
| 20 | SMITH | 800.0 | 5 |
| 30 | BLAKE | 2850.0 | 1 |
| 30 | ALLEN | 1600.0 | 2 |
| 30 | TURNER | 1500.0 | 3 |
| 30 | MARTIN | 1250.0 | 4 |
| 30 | WARD | 1250.0 | 5 |
| 30 | JAMES | 950.0 | 6 |
+------------+-------+------------+------------+
```

1.6.7.4.16. CLUSTER_SAMPLE

Command syntax:

```
boolean cluster_sample(bigint x[, bigint y]) over(partition by col1[, col2..])
```

Purpose: It is used to conduct cluster sampling.

Description:

- x: bigint type. x>=1. If the parameter y is specified, x indicates that a window is divided into x parts. Otherwise, x indicates that x rows of records are extracted from a window (that is, the returned value is true if there are x rows). If x is NULL, NULL is returned.

- y: a constant of the bigint type. y>=1, y<=x. This parameter extracts y records from x parts into which a window is divided (that is, the returned value is true if y records exist). If y is NULL, NULL is returned.
- partition by col1[, col2]: specifies the partitions used in the computation.

Returned value: boolean type.

Example:

The test_tbl table has two columns: key and value. The key column stores the group name of each value. The group names are groupa and groupb. The value column stores the values. The table structure is like this:

```
+------------+--------------------+
| key        | value              |
+------------+--------------------+
| groupa | -1.34764165478145 |
| groupa | 0.740212609046718 |
| groupa | 0.167537127858695 |
| groupa | 0.630314566185241 |
| GroupA | 0.0112401388646925 |
| groupa | 0.199165745875297 |
| groupa | -0.320543343353587 |
| groupa | -0.273930924365012 |
| groupa | 0.386177958942063 |
| groupa | -1.09209976687047 |
| groupb | -1.10847690938643 |
| groupb | -0.725703978381499 |
| groupb | 1.05064697475759 |
| groupb | 0.135751224393789 |
| groupb | 2.13313102040396 |
| groupb | -1.11828960785008 |
| groupb | -0.849235511508911 |
| groupb | 1.27913806620453 |
| groupb | -0.330817716670401 |
| groupb | -0.300156896191195 |
| groupb | 2.4704244205196 |
| groupb | -1.28051882084434 |
+------------+--------------------+
```

Run the following SQL statement to take a sample of 10% of the values in each group:

```
select key, value from (select key, value, cluster_sample(10, 1) over(partition by key) as flag from tbl)
sub where flag = true;
-- Returned result:
+--------+--------------------+
| key    | value              |
+--------+--------------------+
| groupa | -0.273930924365012 |
| groupb | -1.11828960785008  |
+-----+-----------------------
```

1.6.7.4.17. NTILE

Function declaration:

```
BIGINT ntile(BIGINT n) over(partition by col1[, col2...]  [order by col1 [asc|desc] [, col2[asc|desc]...]] [wind
owing_clause]))
```

Purpose: It is used to split grouped data into n slices and return the current slice number. If the slice is uneven, the distribution of the first slice is increased.

Description:

n: BIGINT type.

Returned value: BIGINT type.

Example:

Table emp has the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Group all employees by department, sort each group in descending order by salary, and then obtain sequence numbers of employees in each group.

```
-- Execute the following statement:
select  deptno, ename,sal,NTILE(3) OVER(PARTITION BY deptno ORDER BY sal desc) AS nt3 from emp;
-- Returned result:
+------------+-------+------------+------------+
| deptno     | ename | sal        | nt3        |
+------------+-------+------------+------------+
| 10         | JACCKA | 5000.0     | 1          |
| 10         | KING  | 5000.0     | 1          |
| 10         | WELAN | 2450.0     | 2          |
| 10         | CLARK | 2450.0     | 2          |
| 10         | TEBAGE | 1300.0     | 3          |
| 10         | MILLER | 1300.0     | 3          |
| 20         | SCOTT | 3000.0     | 1          |
| 20         | FORD  | 3000.0     | 1          |
| 20         | JONES | 2975.0     | 2          |
| 20         | ADAMS | 1100.0     | 2          |
| 20         | SMITH | 800.0      | 3          |
| 30         | BLAKE | 2850.0     | 1          |
| 30         | ALLEN | 1600.0     | 1          |
| 30         | TURNER | 1500.0     | 2          |
| 30         | MARTIN | 1250.0     | 2          |
| 30         | WARD  | 1250.0     | 3          |
| 30         | JAMES | 950.0      | 3          |
+------------+-------+------------+------------+
```

1.6.7.4.18. NTH_VALUE

Function declaration:

```
nth_value(expr, bigint n [, boolean skipNulls]) over(partition by col1[, col2...] order by col1 [asc|desc][, c
ol2[asc|desc]...]])
```

Purpose: It is used to return the nth value in partitions used in the computation.

Description:

- expr: required. Any type.
- n: returns the nth value. It starts from 1 and is of the BIGINT type.
- skipNulls: specifies whether to ignore the rows whose values are NULL. This parameter is of the BOOLEAN type. The default value is false.

Returned value: the nth value in partitions used in the computation.

> **Note** If skipNulls is set to true, the nth non-NULL value is returned. If the nth non-NULL value does not exist, NULL is returned.

Example:

```
select a, nth_value(a + 1, 1) over (partition by a order by a) from values (3), (1), (2) as t(a);
-- If n is 1, NTH_VALUE is equivalent to FIRST_VALUE.
-- Returned results:
-- 1  2
-- 2  3
-- 3  4
```

1.6.7.4.19. CUME_DIST

Function declaration:

```
cume_dist() over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...]])
```

Purpose: It is used to return the cumulative distribution. The cumulative distribution is the ratio between the number of rows whose values are less than or equal to the current value of the group and the total number of rows in the group.

Description: None.

> **Note** The order by column specifies values to be compared.

Returned value: the ratio of the number of rows whose values are equal to or less than the current value in the group to the total number of rows in the group.

Example:

Table emp has the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Group all employees by department, and then obtain the cumulative distribution of salary for each group.

```
SELECT deptno
   , ename
, sal
, concat(round(cume_dist() OVER(PARTITION BY deptno ORDER BY sal desc)*100,2),'%') as cume_dist
FROM emp;
```

Returned result is as follows.

Returned result

deptno	ename	sal	cume_dist
10	JACCKA	5000.0	33.33%
10	KING	5000.0	33.33%
10	CLARK	2450.0	66.67%
10	WELAN	2450.0	66.67%
10	TEBAGE	1300.0	100.0%
10	MILLER	1300.0	100.0%

deptno	ename	sal	cume_dist
20	SCOTT	3000.0	40.0%
20	FORD	3000.0	40.0%
20	JONES	2975.0	60.0%
20	ADAMS	1100.0	80.0%
20	SMITH	800.0	100.0%
30	BLAKE	2850.0	16.67%
30	ALLEN	1600.0	33.33%
30	TURNER	1500.0	50.0%
30	MARTIN	1250.0	83.33%
30	WARD	1250.0	83.33%
30	JAMES	950.0	100.0%

1.6.7.4.20. FIRST_VALUE

Function declaration:

```
first_value(expr) over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...]])
```

Purpose: It is used to sort partitions and return the first value in the range from the beginning to the current row.

Description:

expr: required. Any type.

Returned value: the first expr value in partitions used in the computation.

Example:

Table emp has the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Group all employees by department, sort each group in descending order by salary, and then obtain the name of the first employee in each group.

```
SELECT deptno
      , ename
      , sal
      , FIRST_VALUE(ename) OVER(PARTITION BY deptno ORDER BY sal desc) AS first1-- Obtain the name of the first employee in each group after descending sorting by salary.
  FROM emp;
```

Returned result is as follows.

Returned result

deptno	ename	sal	first1
10	JACCKA	5000.0	JACCKA
10	KING	5000.0	JACCKA
10	CLARK	2450.0	JACCKA
10	WELAN	2450.0	JACCKA
10	TEBAGE	1300.0	JACCKA

deptno	ename	sal	first1
10	MILLER	1300.0	JACCKA
20	SCOTT	3000.0	SCOTT
20	FORD	3000.0	SCOTT
20	JONES	2975.0	SCOTT
20	ADAMS	1100.0	SCOTT
20	SMITH	800.0	SCOTT
30	BLAKE	2850.0	BLAKE
30	ALLEN	1600.0	BLAKE
30	TURNER	1500.0	BLAKE
30	MARTIN	1250.0	BLAKE
30	WARD	1250.0	BLAKE
30	JAMES	950.0	BLAKE

1.6.7.4.21. LAST_VALUE

Function declaration:

```
last_value(expr) over(partition by col1[, col2...] order by col1 [asc|desc][, col2[asc|desc]...]])
```

Purpose: It is used to sort partitions and return the last value in the range from the beginning to the current row.

Description:

expr: required. Any type.

Returned value: the last expr value in partitions used in the computation.

Example:

Table emp has the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Group all employees by department, and then obtain the name of the last employee in each group.

```
SELECT deptno
      , ename
      , sal
      , LAST_VALUE(ename) OVER(PARTITION BY deptno  ) AS last1
  FROM emp;
```

The returned result is as follows.

Returned result

deptno	ename	sal	last1
10	TEBAGE	1300.0	WELAN
10	CLARK	2450.0	WELAN
10	KING	5000.0	WELAN
10	MILLER	1300.0	WELAN
10	JACCKA	5000.0	WELAN
10	WELAN	2450.0	WELAN

deptno	ename	sal	last1
20	FORD	3000.0	JONES
20	SCOTT	3000.0	JONES
20	SMITH	800.0	JONES
20	ADAMS	1100.0	JONES
20	JONES	2975.0	JONES
30	TURNER	1500.0	BLAKE
30	JAMES	950.0	BLAKE
30	ALLEN	1600.0	BLAKE
30	WARD	1250.0	BLAKE
30	MARTIN	1250.0	BLAKE
30	BLAKE	2850.0	BLAKE

1.6.7.5. Aggregate functions

1.6.7.5.1. Overview

An aggregate function aggregates multiple input records into an output record. The input is mapped many-to-one to the output. An aggregate function can be used with the GROUP BY clause at the same time.

1.6.7.5.2. COUNT

Command syntax:

```
bigint count([distict|all] value)
```

Purpose: It is used to return the number of records.

Description:

- distinct|all: indicates whether duplicate records are cleared in counting. The default value is all, indicating that records are counted. If it is set to distinct, only records with distinct values are counted.
- value: any type. When it is NULL, this row is not involved in computation. value can be *. When it is set to count(*), the number of all rows is returned.

Returned value: bigint type.

Example:

In the tbla table, the col1 column is of the bigint type.

```
+------+
| COL1 |
+------+
| 1    |
+------+
| 2    |
+------+
| NULL |
+------+
select count(*) from tbla;
-- 3 is returned.
select count(col1) from tbla;
-- The value is 2.
```

Aggregate functions can be used with the GROUP BY statement. For example, table test_src contains two columns: key (string type), and value (double type).

The data in the test_src table:

```
+-----+-------+
| key | value |
+-----+-------+
| a   | 2.0   |
+-----+-------+
| a   | 4.0   |
+-----+-------+
| b   | 1.0   |
+-----+-------+
| b   | 3.0   |
+-----+-------+
select key, count(value) as count from test_src group by key;
-- Run the preceding SQL statement. The output is:
+-----+-------+
| key | count |
+-----+-------+
| a   | 2     |
+-----+-------+
| b   | 2     |
+-----+-------+
```

Aggregate functions perform aggregation on values of the same key. The usage of the following aggregate functions is the same as that of this function and is not described in detail in this document.

1.6.7.5.3. AVG

Function declaration:

```
double avg(double value)
decimal avg(decimal value)
```

Purpose: It is used to calculate the average value.

Description:

value: double type or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned. If the value is NULL, this row is not used for calculation. The input cannot be of the boolean type.

Returned value: If the input is of the decimal type, a value of the decimal type is returned. For all other valid input types, a value of the double type is returned.

Example:

In the tbla table, the value column is of the bigint type.

```
+-------+
| value |
+-------+
| 1     |
| 2     |
| NULL  |
+-------+
select avg(value) as avg from tbla;
+------+
| avg  |
+------+
| 1.5  |
+------+
-- The avg result of this column is as follows: (1 + 2) / 2 = 1.5.
```

1.6.7.5.4. MAX

Function declaration:

```
max(value)
```

Purpose: It is used to return the maximum value.

Description:

value: can be any data type. If the column value is NULL, the corresponding row is not involved in the operation. Values of the boolean type are excluded from the computation.

Returned value: The type is the same as that of value.

Example:

In the tbla table, the col1 column is of the bigint type.

```
+------+
| col1 |
+------+
| 1    |
+------+
| 2    |
+------+
| NULL |
+------+
select max(value) from tbla;
-- 2 is returned.
```

1.6.7.5.5. MIN

Function declaration:

```
MIN(value)
```

Purpose: It is used to return the minimum value.

Description:

value: a column of any data type. If a value in the column is NULL, the corresponding row is not involved in the operation. Boolean types are not allowed in this operation.

Returned value: The type is the same as that of value.

Example:

In the tbla table, the value column is of the bigint type.

```
+------+
| value|
+------+
| 1    |
+------+
| 2    |
+------+
| NULL |
+------+
select min(value) from tbla;
-- 1 is returned.
```

1.6.7.5.6. MEDIAN

Function declaration:

```
double median(double number)
decimal median(decimal number)
```

Purpose: It is used to calculate the median.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other input types, an error is returned. If the input is NULL, a failure is returned.

Returned value: double or decimal type.

1.6.7.5.7. STDDEV

Function declaration:

```
double stddev(double number)
decimal stddev(decimal number)
```

Purpose: It is used to calculate the population standard deviation.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned. If the input value is NULL, a failure is returned.

Returned value: double or decimal type.

1.6.7.5.8. STDDEV_SAMP

Function declaration:

```
double stddev_samp(double number)
decimal stddev_samp(decimal number)
```

Purpose: It is used to calculate the sample standard deviation.

Description:

number: double or decimal type. If the input is of the string or bigint type, it is implicitly converted into a value of the double type before this computation. For all other types of inputs, an error is returned. If the input is NULL, a failure is returned.

Returned value: double or decimal type.

1.6.7.5.9. SUM

Function declaration:

```
sum(value)
```

Purpose: It is used to calculate the sum.

Description:

value: double, decimal, or bigint type. If the input is of the string type, it is implicitly converted into a value of the double type before computation. If a value in the column is NULL, this row is not used for calculation. Values of the boolean type are excluded from calculation.

Returned value: When the input is of the bigint type, a value of the bigint type is returned. When the input is of the double or string type, a value of the double type is returned.

Example:

In the tbla table, the value column is of the bigint type.

```
+------+
| value|
+------+
| 1    |
+------+
| 2    |
+------+
| NULL |
+------+
select sum(value) from tbla;
-- 3 is returned.
```

1.6.7.5.10. WM_CONCAT

Function declaration:

```
string wm_concat(string separator, string str)
```

Purpose: It is used to use the specified separator as the delimiter to link values in a string.

Description:

- separator: the delimiter, which is a constant of the string type. If it is of another type or is not a constant, an error is returned.
- str: string type. If the input is of the bigint, double, or datetime type, it is implicitly converted to a value of the string type before this computation. For all other input types, an error is returned.

Returned value: string type.

> **Note** If test_src in the select wm_concat(',', name) from > test_src; statement is an empty set, NULL is returned.

1.6.7.5.11. PERCENTILE

Function declaration:

```
DOUBLE percentile(BIGINT col, p)
array<double> percentile(BIGINT col, array(p1 [, p2]...))
```

Purpose: It is used to return the pth percentile of the specified column. p must be between 0 and 1.

> **Notice** You can only calculate true percentiles for integer values.

Description:

- col: BIGINT type.
- p: must be between 0 and 1.

Example:

Column c1 in table test has the following data:

```
+------------+
| c1       |
+------------+
| 8        |
| 9        |
| 10       |
| 11       |
+------------+
```

Calculate the pth percentile of column c1 in table test.

```
-- Execute the following statement:
select percentile(c1,0),percentile(c1,0.3),percentile(c1,0.5),percentile(c1,1) from test;
-- Returned result:
+------------+------------+------------+------------+
| _c0      | _c1      | _c2      | _c3      |
+------------+------------+------------+------------+
| 8.0      | 8.9      | 9.5      | 11.0     |
+------------+------------+------------+------------+
-- Execute the following statement:
select percentile(c1,array(0,0.3,0.5,1))from test;
-- Returned result:
+------+
| _c0  |
+------+
| [8, 8.9, 9.5, 11] |
+------+
```

1.6.7.5.12. Additional aggregate functions

MaxCompute 2.0 provides additional aggregate functions. You must add the following SET statement before SQL statements contained in the aggregate functions:

```
set odps.sql.type.system.odps2=true;
```

> **Note** You must submit and execute the SET statement and the SQL statements of the new functions simultaneously.

The aggregate functions described in subsequent topics are new in MaxCompute 2.0.

1.6.7.5.13. COLLECT_LIST

Command syntax:

```
ARRAY collect_list(col)
```

Purpose: It is used to convert the values on the col column into an array.

Description:

col: a table column of any data type.

Returned value: array type.

1.6.7.5.14. COLLECT_SET

Command syntax:

```
ARRAY collect_set(col)
```

Purpose: It is used to convert the values on the col column with duplicates removed into an array.

Description:

col: a table column of any data type.

Returned value: array type.

1.6.7.5.15. VARIANCE/VAR_POP

Function declaration:

```
DOUBLE variance(col)
DOUBLE var_pop(col)
```

Purpose: It is used to calculate the variance of the specified numeric column.

Description:

col: numeric type column. NULL is returned for other types.

Returned value: DOUBLE type.

Example:

Column c1 in table test has the following data:

```
+------------+
| c1       |
+------------+
| 8        |
| 9        |
| 10       |
| 11       |
+------------+
```

Calculate the variance of column c1 in table test.

```
-- Execute the following statement:
select variance(c1) from test;
-- or
select var_pop(c1) from test;
-- Returned result:
+------------+
| _c0        |
+------------+
| 1.25       |
+------------+
```

1.6.7.5.16. VAR_SAMP

Function declaration:

```
DOUBLE var_samp(col)
```

Purpose: It is used to calculate the sample variance of the specified numeric column.

Description:

col: numeric type column. NULL is returned for other types.

Returned value: DOUBLE type.

Example:

Column c1 in table test has the following data:

```
+------------+
| c1         |
+------------+
| 8          |
| 9          |
| 10         |
| 11         |
+------------+
```

Calculate the variance of column c1 in table test.

```
-- Execute the following statement:
select var_samp(c1) from test;
-- Returned result:
+------------+
| _c0       |
+------------+
| 1.6666666666666667 |
+------------+
```

1.6.7.5.17. COVAR_POP

Function declaration:

```
DOUBLE covar_pop(col1, col2)
```

Purpose: It is used to calculate the population covariance of two specified numeric columns.

Description:

col1 and col2: numeric type columns. NULL is returned for other types.

Example:

Columns c1 and c2 in table test have the following data:

```
+------------+------------+
| c1        | c2        |
+------------+------------+
| 3         | 2         |
| 14        | 5         |
| 50        | 14        |
| 26        | 75        |
+------------+------------+
```

Calculate the population covariance of columns c1 and c2.

```
-- Execute the following statement:
select covar_pop(c1,c2) from test;
-- Returned result:
+------------+
| _c0       |
+------------+
| 123.49999999999997|
+------------+
```

1.6.7.5.18. COVAR_SAMP

Function declaration:

```
DOUBLE covar_samp(col1, col2)
```

Purpose: It is used to calculate the sample covariance of two specified numeric columns.

Description:

col1 and col2: numeric type columns. NULL is returned for other types.

Example:

Columns c1 and c2 in table test have the following data:

```
+------------+------------+
| c1         | c2         |
+------------+------------+
| 3          | 2          |
| 14         | 5          |
| 50         | 14         |
| 26         | 75         |
+------------+------------+
```

Calculate the sample covariance of columns c1 and c2.

```
-- Execute the following statement:
select covar_samp(c1,c2) from test;
-- Returned result:
+------------+
| _c0        |
+------------+
| 164.66666666666663|
+------------+
```

1.6.7.6. Other functions

1.6.7.6.1. ARRAY

Function declaration:

```
array(value1,value2, ...)
```

Purpose: It is used to create an array by using input values.

Description:

value: any type. All the values must be of the same type.

Returned value: ARRAY type.

Example:

```
select array(123,456,789) from dual;
-- Returned result:
[123, 456, 789]
```

1.6.7.6.2. ARRAY_CONTAINS

Function declaration:

```
array_contains(ARRAY<T> a, value v)
```

Purpose: It is used to check whether array a contains value v.

Description:

- a: array type.
- v: The given value v must be of the same type as the data in the array.

Returned value: boolean type.

Example:

```
select array_contains(array('a','b'), 'a') from dual;
-- True is returned.
select array_contains(array(456,789),123) from dual;
-- False is returned.
```

1.6.7.6.3. CAST

Command syntax:

```
cast(expr as <type>)
```

Purpose: It is used to convert an expression of one data type to another. For example, cast ('1' as bigint) converts 1 of the string type to the integer type. If the conversion fails, an error is returned.

Note

- cast(double as bigint) converts a value of the double type into a value of the bigint type.
- cast(string as bigint) converts a value of the string type into a value of the bigint type. If the string is composed of numerals expressed in integer form, it is directly converted into a value of the bigint type. If the string is comprised of numerals expressed in the 'float' or 'exponent' form, it is converted to 'double' type first and then to 'bigint' type.
- For cast(string as datetime) or cast(datetime as > string), the datatime format is yyyy-mm-dd hh:mi:ss by default.

1.6.7.6.4. COALESCE

Command syntax:

```
coalesce(expr1, expr2, ...)
```

Purpose: It is used to return the first non-NULL value in the list. If all values in the list are NULL, NULL is returned.

Description:

expr: a value to be tested. All these values must be of the same type or be NULL. Otherwise, an error is returned.

Returned value: The type is the same as that of the input.

Note At least one parameter is provided. Otherwise, an error is returned.

1.6.7.6.5. DECODE

Function declaration:

```
decode(expression, search, result[, search, result]...[, default])
```

Purpose: It is used to implement the if-then-else conditional branching feature.

Description:

- expression: expression to be compared.
- search: search string to be compared with the expression.
- result: the value returned when the value of search matches the expression.
- default: optional. If no search string matches the expression, the default value is returned. If it is not specified, NULL is returned.

Returned value: The matched search is returned. If there are no matches, the default value is returned. If default is not specified, NULL is returned.

Note

- At least three parameters are specified.
- All results must share the same type or be NULL. Inconsistent data types will cause an error. All values of search and expression must be of the same type. Otherwise, an error is returned.
- If the search option in decode has repeated records and matches the expression, the first search value is returned.

Example:

```
select decode(customer_id,
1, 'Taobao',
2, 'Alipay',
3, 'Aliyun',
NULL, 'N/A',
'Others') as result from sale_detail;
```

The preceding DECODE function implements the feature in the following if-then-else statement:

```
if customer_id = 1 then result := 'Taobao';
elsif customer_id = 2 then result := 'Alipay';
elsif customer_id = 3 then result := 'Aliyun';
...
else
result := 'Others';
end if;
```

Notice The MaxCompute SQL statement returns NULL when calculating NULL = NULL. However, in the DECODE function, values of NULL and NULL are equal. In the preceding example, when the value of customer_id is NULL, the DECODE function returns N/A.

1.6.7.6.6. EXPLODE

Function declaration:

```
explode (var)
```

Purpose: It is used to convert one row of data into multiple rows of UDTF. If var is of the array type, the array stored in the column is converted into multiple rows. If var is of the map type, each key-value pair of the map stored in the column is converted into a row with two columns, with one column for the key and the other for the value.

Description:

var: array < T > type or map < K,V > type.

Returned value: transposed rows.

> **Note**
>
> Limits on the use of UDTFs:
>
> - Only one UDTF is allowed in a SELECT statement, and other columns are not allowed.
> - One select can only have one UDTF and no other columns can appear.

Example:

```
explode(array(null, 'a', 'b', 'c'))  col
```

1.6.7.6.7. GET_IDCARD_AGE

Function declaration:

```
get_idcard_age(idcardno)
```

Purpose: It is used to return the current age based on the ID card number. The current age is the current year minus the birth year on the ID card.

Description:

idcardno: string type, ID number of 15-digit or 18-digit. During the calculation, the validity of the ID card is verified based on the province code and the last check code. If the verification fails, NULL is returned.

Returned value: bigint type. If the input is NULL, NULL is returned. If the difference of the current year minus the birth year is greater than 100, then NULL is returned.

1.6.7.6.8. GET_IDCARD_BIRTHDAY

Function declaration:

```
get_idcard_birthday(idcardno)
```

Purpose: It is used to return the date of birth based on the ID card number.

Description:

idcardno: string type, a 15-digit or 18-digit ID card number. During computation, the validity of the ID card is verified based on the province code and the last check code. If the verification fails, NULL is returned.

Returned value: datetime type. If the input is NULL, NULL is returned.

1.6.7.6.9. GET_IDCARD_SEX

Function declaration:

```
get_idcard_sex(idcardno)
```

Purpose: It is used to return the gender based on the ID card number. The returned value is M (male) or F (female).

Description:

idcardno: string type, a 15-digit or 18-digit ID card number. During computation, the validity of the ID card is verified based on the province code and the last check code. If the verification fails, NULL is returned.

Returned value: string type. If the input is NULL, NULL is returned.

1.6.7.6.10. GREATEST

Function declaration:

```
greatest(var1, var2, ...)
```

Purpose: It is used to return the maximum value among the input values.

Description:

var: bigint, double, datetime, or string type. If all values are NULL, NULL is returned.

Returned value:

- The greatest value in input parameter. If the implicit conversion is not needed, return type is the same as input parameter type.
- NULL is interpreted as the minimum value.
- If the input parameters are of different types, values of the double, bigint, and string types are converted into values of the double type for comparison, and values of the string and datetime types are converted into values of the datetime type for comparison. Implicit conversion of other types is not allowed.

1.6.7.6.11. INDEX

Function declaration:

```
index(var1[var2])
```

Purpose: It is used to return the specified element in a given array, or return the value of the specified key in a given map.

Description:

- var1: array < T > type or map < K,V > type.
- var2: If var1 is of the array < T > type, var2 must be the bigint type must be larger or equal to 0. If var1 is of the map < K,V > type, var2 is of the K type.

Returned value:

- If var1 is of the array < T > type, a value of the T type is returned. If var2 is out of range of array < T > elements, NULL is returned.

- If var1 is of the map < K,V > type, a value of the V type is returned. If no key is var2 in map < K,V >, NULL is returned.

Example:

If var1 is an array, run the following SQL statement:

```
select array('a','b','c')[2] from dual;
-- Returned result:
+-----+
| _c0 |
+-----+
| c   |
+-----+
```

If var1 is of the map type, run the following SQL statement:

```
select str_to_map("test1=1,test2=2")["test1"] from dual;
-- Returned result:
+-----+
| _c0 |
+-----+
| 1   |
+-----+
```

> **Notice**
> - To use the SQL statement, remove the index and run var1[var2] directly. Otherwise, a syntax error is returned.
> - If Var1 is NULL, NULL is returned.

1.6.7.6.12. MAX_PT

Function declaration:

```
max_pt(table_full_name)
```

Purpose: For partitioned tables, it is used to return the maximum values in the first-level partitions that have data files and sort the values in alphabetic order.

Description:

table_full_name: string type. It specifies a table name (project name required, for example, prj.src). You must have the read permission on the table.

Returned value: maximum value in the primary partition.

Example:

Partitioned table tbl has the following partitions with data files: pt='20170901' and pt='20170902'. In the following statement, the returned value of max_pt is '20170902'. The MaxCompute SQL statement reads data from the '20120902' partition.

```
select * from tbl where pt=max_pt('myproject.tbl');
```

Note If a new partition is added by using alter table, but there is no data file in this partition, then this partition is not returned.

1.6.7.6.13. ORDINAL

Function declaration:

```
ordinal(bigint nth, var1, var2, ...)
```

Purpose: It is used to sort the input variables in ascending order, and return the specified nth value.

Description:

- nth: bigint type. It specifies the position at which the value is to be returned. If it is NULL, NULL is returned.
- var: bigint, double, datetime, or string type.

Returned value:

- The value in nth bit. If the implicit conversion is not needed, return type is the same as input parameter type.
- If type conversion is performed, values of the double, bigint, and string types are converted into values of the double type. Values of the string and datetime types are converted into values of the datetime type. Implicit conversion of other types is not allowed.
- NULL is the least value.

Example:

```
ordinal(3, 1, 3, 2, 5, 2, 4, 6) =  2
```

1.6.7.6.14. LEAST

Function declaration:

```
least(var1, var2, ...)
```

Purpose: It is used to returns the minimum value among the input values.

Description:

var: bigint, double, datetime, or string type. If all values are NULL, NULL is returned.

Returned value:

- The least value in input parameter; If the implicit conversion is not needed, return type is the

same as input parameter type.

- If type conversion is performed, values of the double, bigint, and string types are converted into values of the double type. Values of the string and datetime types are converted into values of the datetime type. Implicit conversion of other types is not allowed.
- NULL is interpreted as the minimum value.

1.6.7.6.15. SIZE

Function declaration:

```
size(map<K, V>)
size(array<T>)
```

Purpose: size(map) is used to return the number of key-value pairs in the given map, and size(array) is used to return the number of elements in the given array.

Description:

- map: map type.
- array: array type.

Returned value: int type.

Example:

```
select size(map('a',123,'b',456)) from dual;
-- 2 is returned.
select size(map('a',123,'b',456,'c',789)) from dual;
-- 3 is returned.
select size(array('a','b')) from dual;
-- 2 is returned.
select size(array(123,456,789)) from dual;
-- 3 is returned.
```

1.6.7.6.16. SPLIT

Function declaration:

```
split(str, pat)
```

Purpose: It is used to split a string using the specified separator.

Description:

- str: string type. The string to be separated.
- pat: string type. It indicates the separator and supports regular expressions.

Returned value: array <string >. The returned array contains elements extracted from the string based on the specified separator.

Example:

```
select split("a,b,c",",") from dual;
-- Returned result:
+------+
| _c0 |
+------+
| [a, b, c] |
+------+
```

1.6.7.6.17. STR_TO_MAP

Function declaration:

```
str_to_map(text [, delimiter1 [, delimiter2]])
```

Purpose: It is used to divide 'text' into K-V pairs with 'delimiter1', and to separate each K-V pair with 'delimiter2'.

Description:

ext: string type. It indicates the string to be separated.

delimiter1: string type. It is the delimiter. If it is not specified, the default value ',' is used.

delimiter2: string type. It is the delimiter. If it is not specified, the default value '=' is used.

Returned value: map < string, string >. The elements are the K-V results of the separation of 'text' by the strings 'delimiter1' and 'delimiter2'.

Example:

```
select str_to_map("test1=1,test2=2") from dual;
-- Returned result:
+------------+
| a          |
+------------+
| {Test1: 1, Test2: 2}|
```

1.6.7.6.18. UNIQUE_ID

Function declaration:

```
STRING UNIQUE_ID()
```

Purpose: It is used to return a random but unique ID, for example, 29347a88-1e57-41ae-bb68-a9edbdd94212_1. This function runs more efficiently than UUID.

1.6.7.6.19. UUID

Function declaration:

```
string uuid()
```

Purpose: It returns a random ID, for example, 29347a88-1e57-41ae-bb68-a9edbdd94212.

1.6.7.6.20. SAMPLE

Function declaration:

```
boolean sample(x, y, column_name)
```

Purpose: It is used to sample all values read from the specified column based on the given settings, and filters out the rows that do not meet the sampling condition.

Description:

- x, y: bigint type. It indicates that data is hashed to x portions and the yth portion is taken. y can be omitted. If y is omitted, the first portion is taken and column_name must also be omitted. x and y are constants of the integer type and are greater than 0. If they are of another type or if they are less than or equal to 0, an error is returned. If y is greater than x, an error is returned. If either x or y is NULL, NULL is returned.
- column_name: target column of sampling. column_name can be omitted. If column_name is omitted, random sampling is performed based on values of x and y. It can be of any type, and the column value can be NULL. No implicit conversion is performed. If column_name is the constant NULL, an error is reported.

Returned value: boolean type.

> **Note** To avoid data skew resulting from the NULL value, a uniform hash of x is made for a value of NULL in column_name. If column_name is not added, the output is not necessarily uniform since the data size is smaller. So column_name is suggested to be added to get better output.

Example:

Table tbla contains a column named cola.

```
select * from tbla where sample (4, 1 , cola) = true;
-- The values are hashed to four portions based on cola, and the first portion is used.
select * from tbla where sample (4, 2) = true;
-- The values in each row are randomly hashed to four portions, and the second portion is used.
```

1.6.7.6.21. CASE WHEN expression

MaxCompute provides the following two kinds of CASE WHEN syntax formats:

```
case value
when (_condition1) then result1
when (_condition2) then result2
...
else resultn
end

case
when (_condition1) then result1
when (_condition2) then result2
when (_condition3) then result3
...
else resultn
end
```

CASE WHEN flexibly returns different values based on the calculation result of the expression. Alibaba Cloud StreamCompute supports two types of CASE WHEN expressions:

```
select
case
when shop_name is null then 'default_region'
when shop_name like 'hang%' then 'zj_region'
end as region
From sale_detail;
```

> **Note**
>
> - If there are values of only the bigint and double type in the results, the results are converted into values of the double type.
> - If there is a value of the string type in the results, the results are all converted into values of the string type. If the result of a type cannot be converted (for example, boolean type), an error is returned.
> - Conversion between other types is not allowed.

1.6.7.6.22. IF

Function declaration:

```
if(testCondition, valueTrue, valueFalseOrNull)
```

Purpose: It is used to determine whether 'testCondition' is true. If it is true, valueTrue is returned. If it is not true, valueFalseOrNull is returned.

Description:

testCondition: boolean type. The expression to be determined true or not.

valueTrue: the value returned when expression 'testCondition' is true.

valueFalseOrNull: the value returned when expression 'testCondition' is false. It can be set to NULL.

Returned value: The type is the same as that of valueTrue or valueFalseOrNull.

Example:

```
select if(1=2,100,200) from dual;
-- Returned result:
+------------+
| _c0        |
+------------+
| 200        |
+------------+
```

1.6.7.6.23. Additional functions

MaxCompute 2.0 provides additional functions.

The functions described in the following topics are new in this version.

1.6.7.6.24. MAP

Function declaration:

```
map(K key1, V value1, K key2, V value2, ...)
```

Purpose: It is used to create a map with the given K-V pairs.

Description:

key/value: The types of all keys are the same and must be of one of the basic types. The types of all values are the same and can be of any type.

Returned value: map type.

Example:

```
select map('a',123,'b',456) from dual;
-- Returned result:
{a:123, b:456}
```

1.6.7.6.25. MAP_KEYS

Function declaration:

```
map_keys(map<K, V> )
```

Purpose: It is used to return all keys in the map parameter as an array.

Description:

map: data of the map type.

Returned value: array type. If the input is NULL, NULL is returned.

Example:

```
select map_keys(map('a',123,'b',456)) from dual;
-- Returned result:
[a, b]
```

1.6.7.6.26. MAP_VALUES

Function declaration:

```
map_values(map<K, V>)
```

Purpose: It is used to return all values in the map parameter as an array.

Description:

map: map type.

Returned value: array type. If the input is NULL, NULL is returned.

Example:

```
select map_keys(map('a',123,'b',456)) from dual;
-- Returned result:
[123, 456]
```

1.6.7.6.27. SORT_ARRAY

Function declaration:

```
sort_array(ARRAY<T>)
```

Purpose: It is used to sort a given array.

Description:

ARRAY: array type. The data in the array is of any type.

Returned value: array type.

Example:

```
select sort_array(array('a','c','f','b')),sort_array(array(4,5,7,2,5,8)),sort_array(array('You','Me','He')) from
dual;
-- Returned result:
[a, b, c, f] [2, 4, 5, 5, 7, 8] [him, you, me]
```

1.6.7.6.28. POSEXPLODE

Command syntax:

```
posexplode(ARRAY<T>)
```

Purpose: It is used to explode the given array. Each value is given a row and each row has two columns corresponding to the subscript (starting from 0) and the array element.

Description:

ARRAY: array type. Data in the array can be of any type.

Returned value: table generation function.

Example:

```
select posexplode(array('a','c','f','b')) from dual;
-- Returned result:
+------------+-----+
| pos        | val |
+------------+-----+
| 0          | a   |
| 1          | c   |
| 2          | f   |
| 3          | b   |
+------------+-----+
```

1.6.7.6.29. STRUCT

Function declaration:

```
struct(value1,value2, ...)
```

Purpose: It is used to create a struct using a given value list.

Description:

value: any type.

Returned value: struct type. The field names of the created struct are col1, col2, and so on.

Example:

```
select struct('a',123,'ture',56.90) from dual;
-- Returned result:
{col1:a, col2:123, col3:true, col4:56.9}
```

1.6.7.6.30. NAMED_STRUCT

Function declaration:

```
named_struct(string name1, T1 value1, string name2, T2 value2, ...)
```

Purpose: It is used to create a struct using a given name-value list.

Description:

- value: any type.
- name: field name of the string type.

Returned value: struct type. The field names of the created struct are name1, name2, and so on.

Example:

```
select named_struct('user_id',10001,'user_name','bob','married','F','weight',63.50) from dual;
-- Returned result:
{user_id:10001, user_name:bob, married:F, weight:63.5}
```

1.6.7.6.31. INLINE

Function declaration:

```
inline(ARRAY<STRUCT<f1:T1, f2:T2, ... >>)
```

Purpose: It is used to expand a struct, with each element corresponding to a row, and each struct element in each row corresponding to a column.

Description:

STRUCT: The values in the array can be of any type.

Returned value: table generation function.

Example:

```
select inline(array(named_struct('user_id',10001,'user_name','bob','married','F','weight',63.50))) from du
al;
-- Returned result:
+------------+-----------+---------+------------+
| user_id    | user_name | married | weight     |
+------------+-----------+---------+------------+
| 10001      | bob       | F       | 63.5       |
+------------+-----------+---------+------------+
```

1.6.7.6.32. BETWEEN AND expression

Command syntax:

```
A [NOT] BETWEEN B AND C
```

If A, B, or C is NULL, then the value is NULL. If A is greater than or equal to B, and less than or equal to C, the value is true. Otherwise, the value is false.

Example:

The emp table contains the following data:

```
| empno | ename | job | mgr | hiredate| sal| comm | deptno |
7369,SMITH,CLERK,7902,1980-12-17 00:00:00,800,,20
7499,ALLEN,SALESMAN,7698,1981-02-20 00:00:00,1600,300,30
7521,WARD,SALESMAN,7698,1981-02-22 00:00:00,1250,500,30
7566,JONES,MANAGER,7839,1981-04-02 00:00:00,2975,,20
7654,MARTIN,SALESMAN,7698,1981-09-28 00:00:00,1250,1400,30
7698,BLAKE,MANAGER,7839,1981-05-01 00:00:00,2850,,30
7782,CLARK,MANAGER,7839,1981-06-09 00:00:00,2450,,10
7788,SCOTT,ANALYST,7566,1987-04-19 00:00:00,3000,,20
7839,KING,PRESIDENT,,1981-11-17 00:00:00,5000,,10
7844,TURNER,SALESMAN,7698,1981-09-08 00:00:00,1500,0,30
7876,ADAMS,CLERK,7788,1987-05-23 00:00:00,1100,,20
7900,JAMES,CLERK,7698,1981-12-03 00:00:00,950,,30
7902,FORD,ANALYST,7566,1981-12-03 00:00:00,3000,,20
7934,MILLER,CLERK,7782,1982-01-23 00:00:00,1300,,10
7948,JACCKA,CLERK,7782,1981-04-12 00:00:00,5000,,10
7956,WELAN,CLERK,7649,1982-07-20 00:00:00,2450,,10
7956,TEBAGE,CLERK,7748,1982-12-30 00:00:00,1300,,10
```

Run the following command to query data where sal is greater than or equal to 1,000 and less than or equal to 1,500:

```
select * from emp where sal BETWEEN 1000 and 1500;
-- Returned result:
+-------+-------+-----+------------+------------+------------+------------+------------+
| empno | ename | job | mgr        | hiredate   | sal        | comm       | deptno     |
+-------+-------+-----+------------+------------+------------+------------+------------+
| 7521  | WARD  | SALESMAN | 7698  | 1981-02-22 00:00:00 | 1250.0     | 500.0      | 30  |
| 7654  | MARTIN | SALESMAN | 7698 | 1981-09-28 00:00:00 | 1250.0     | 1400.0     | 30 |
| 7844  | TURNER | SALESMAN | 7698 | 1981-09-08 00:00:00 | 1500.0     | 0.0        | 30 |
| 7876  | ADAMS | CLERK | 7788  | 1987-05-23 00:00:00 | 1100.0     | NULL     | 20   |
| 7934  | MILLER | CLERK | 7782  | 1982-01-23 00:00:00 | 1300.0     | NULL      | 10  |
| 7956  | TEBAGE | CLERK | 7748  | 1982-12-30 00:00:00 | 1300.0     | NULL      | 10  |
+-------+-------+-----+------------+------------+------------+------------+------------+
```

1.6.7.6.33. NVL

Function declaration:

```
nvl(T value, T default_value)
```

Purpose: It is used to return default_value if value is NULL and return value otherwise.

Example:

Table t_data has three columns of c1 string, c2 bigint, and c3 datetime, as well as the following data:

```
+----+------------+------------+
| c1 | c2         | c3         |
+----+------------+------------+
| NULL | 20         | 2017-11-13 05:00:00 |
| ddd | 25         | NULL       |
| bbb | NULL       | 2017-11-12 08:00:00 |
| aaa | 23         | 2017-11-11 00:00:00 |
+----+------------+------------+
```

Use the NVL function to output the NULL values in c1 to 00000, the NULL values in c2 to 0, and the NULL values in c3 to "-".

```
-- Execute the following statement:
SELECT nvl(c1,'00000'),nvl(c2,0) nvl(c3,'-') from nvl_test;
-- Returned result:
+-----+------------+-----+
| _c0 | _c1       | _c2 |
+------------+---------+
| bbb | 0         | 2017-11-12 08:00:00 |
| ddd | 25        | -   |
| 00000 | 20        | 2017-11-13 05:00:00 |
| aaa | 23        | 2017-11-11 00:00:00 |
+-----+------------+-----+
```

1.6.7.6.34. TABLE_EXISTS

Function declaration:

```
boolean table_exists(string table_name)
```

Description: This function checks whether a specific table exists.

Parameters:

table_name: the table name of the STRING type. The value can include the project name, such as my_proj.my_table. If no project name is specified, the name of the current project is used.

Return value: A value of the BOOLEAN type is returned. If the specified table exists, True is returned. Otherwise, False is returned.

Example:

```
-- Used in a SELECT statement.
SELECT IF(table_exists('abd'), col1, col2) FROM src;
-- Used in an IF-ELSE branch statement.
IF (table_exists('abd'))
-- statments
ELSE
-- statments
```

1.6.7.6.35. PARTITION_EXISTS

Function declaration:

```
boolean partition_exists(string table_name, string... partitions)
```

Description: This function checks whether a specific partition exists.

Parameters:

- table_name: the table name of the STRING type. The value can include the project name, such as my_proj.my_table. If no project name is specified, the name of the current project is used.
- partitions: the partition names of the STRING type. Set this parameter to the partitioning column values based on partition key columns in sequence. The number of partition names must be the same as that of partition key columns.

Return value: A value of the BOOLEAN type is returned. If the specified partitions exist, True is returned. Otherwise, False is returned.

Example:

```
CREATE TABLE foo (id BIGINT) PARTITIONED BY (ds STRING, hr STRING);
-- Create a partitioned table named foo.
ALTER TABLE foo ADD PARTITION (ds='20190101', hr='1');
-- Add a partition to foo.
SELECT partition_exists('foo', '20190101', '1');
-- Check whether partitions ds='20190101' and hr='1' exist.
```

1.6.8. UDFs

1.6.8.1. Overview

UDF is short for user defined function. MaxCompute provides a variety of built-in functions. You can also create UDFs based on specific computing requirements. You can use UDFs as using common built-in functions. This topic briefs how to use SQL UDFs. For more information about SQL UDFs, see the official documentation on UDFs.

The following table lists the extended UDFs in MaxCompute.

UDF category

UDF category	Description
UDF	User defined scalar functions are commonly referred to as UDFs. There is a one-to-one mapping between the input and output. Each time a UDF reads a row of data, it writes an output value.
UDTF	User defined table valued functions are commonly referred to as UDTFs. Each time a UDTF is called, it outputs multiple rows of data. UDTFs are the only category that returns multiple fields. A UDF only returns one value each time.
UDAF	User defined aggregation functions are commonly referred to as UDAFs. A UDAF aggregates multiple input records into one output record. There is a many-to-one mapping between input and output. A UDAF can be used together with the GROUP BY clause (SQL) at the same time. For more information about the syntax, see aggregation functions.

Note In general, UDFs refer to all user defined functions: UDFs, UDAFs, and UDTFs. In a narrow sense, UDFs only refer to user defined scalar functions. This term is used interchangeably in this document. You will have to determine the exact meaning based on the context.

1.6.8.2. Types of parameters and returned values

UDFs support the following MaxCompute SQL data types:

- Basic data types: BIGINT, DOUBLE, BOOLEAN, DATETIME, DECIMAL, STRING, TINYINT, SMALLINT, INT, FLOAT, VARCHAR, BINARY, and TIMESTAMP.
- Complex data types: ARRAY, MAP, and STRUCT.

Note In UDFs, you can define the writable attribute of parameters.

The usage of some basic data types (such as TINYINT, SMALLINT, INT, FLOAT, VARCHAR, BINARY, and TIMESTAMP) in Java UDFs is as follows:

- UDAFs and UDTFs use the @Resolve annotation to obtain signatures. Example: `@Resolve("smallint->varchar(10)")`.
- UDFs reflect and analyze the evaluate() method to obtain signatures. In this case, there is a one-to-one mapping between MaxCompute built-in types and Java types.

To use complex data types (ARRAY, MAP, and STRUCT) in Java UDFs, take the following steps:

- UDTFs use the @Resolve annotation to specify signatures. Example: `@Resolve("array<string>,struct<a1:bigint,b1:string>,string->map<string,bigint>,struct<b1:bigint>")`.
- UDFs use the signature of the evaluate() method to map the input and output types. For more information, see the mappings between MaxCompute types and Java types. In the preceding example, ARRAY corresponds to java.util.List, MAP corresponds to java.util.Map, and STRUCT corresponds to com.aliyun.odps.data.Struct.
- UDAFs and UDTFs use the @Resolve annotation to obtain signatures. Example: `@Resolve("smallint->varchar(10)")`.

Notice

- You can use `type,*` to add any number of parameters. Example: @resolve("string,*->array<string>"). Note that you must add a subtype after array.
- The field name and field type of com.aliyun.odps.data.Struct cannot be reflected. Therefore, the @Resolve annotation is required. If you want to use struct in a UDF, you must add the @Resolve annotation to the UDF class. This annotation only affects the overloads of parameters or returned values that contain com.aliyun.odps.data.Struct.
- A class supports only one @Resolve annotation. A UDF that contains struct can only reload parameters or returned values once.

The following table lists the mapping between MaxCompute and Java data types.

Data type mapping

MaxCompute type	Java type
TINYINT	java.lang.Byte
SMALLINT	java.lang.Short
INT	java.lang.Integer
BIGINT	java.lang.Long
FLOAT	java.lang.Float
DOUBLE	java.lang.Double
DECIMAL	java.math.BigDecimal
BOOLEAN	java.lang.Boolean
STRING	java.lang.String
VARCHAR	com.aliyun.odps.data.Varchar
BINARY	com.aliyun.odps.data.Binary
DATETIME	java.util.Date
TIMESTAMP	java.sql.Timestamp
ARRAY	java.util.List
MAP	java.util.Map
STRUCT	com.aliyun.odps.data.Struct

Note

- Java data types and the data types of returned values are objects, and must start with a capitalized letter.
- The NULL value in SQL is represented by a NULL reference in Java. The Java primitive type is not allowed because it cannot represent a NULL value in SQL.
- The ARRAY type in MaxCompute corresponds to a list, not an array, in Java.

The following table compares the API features of two languages.

API feature comparison

Supported language	UDF	UDAF	UDTF	DATETIME type	Read resource file	Read resource table
Python	Yes	Yes	Yes	Yes	Yes	Yes

Supported language	UDF	UDAF	UDTF	DATETIME type	Read resource file	Read resource table
Java	Yes	Yes	Yes	Yes	Yes	Yes

1.6.8.3. UDFs

A UDF must inherit the com.aliyun.odps.udf.UDF class and implement the EVALUATE method. The EVALUATE method must be a non-static public method. The types of parameters and returned values of the EVALUATE method are used as the UDF signatures in SQL. This means that users can implement multiple EVALUATE methods in a UDF. When a UDF is called, the framework matches the correct EVALUATE method based on the parameter type called by the UDF.

Example:

```
package org.alidata.odps.udf.examples;
import com.aliyun.odps.udf.UDF;
public final class Lower extends UDF { public String evaluate(String s) { if (s == null) { return null; } retu
rn s.toLowerCase();
}
}
```

Note You can implement `void setup(ExecutionContext ctx)` and `void close()` to implement UDF initialization and termination code, respectively.

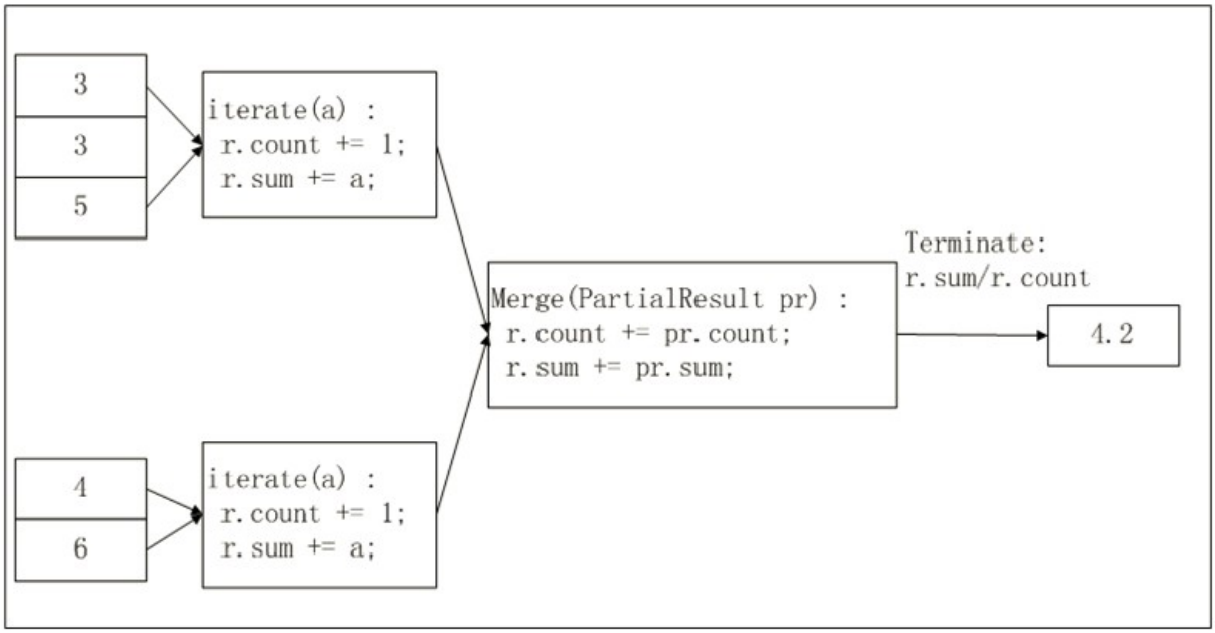
UDFs are used in the same way as built-in functions in MaxCompute SQL. For more information, see Built-in functions.

1.6.8.4. UDAFs

To implement a Java UDAF, you must inherit the com.aliyun.odps.udf.UDAF class and implement the following APIs:

```
public abstract class Aggregator implements ContextFunction {
@Override
public void setup(ExecutionContext ctx) throws UDFException {
}
@Override
public void close() throws UDFException {
}
/**
* Create an aggregate buffer
* @return Writable - Aggregate buffer
*/
abstract public Writable newBuffer();
/**
* @param buffer - Aggregate buffer
* @param args - Parameter specified when SQL calls UDAFs
* @throws UDFException
*/
abstract public void iterate(Writable buffer, Writable[] args) throws UDFException;
/**
* generate final result
* @param buffer
* @return final result of Object UDAF
* @throws UDFException
*/
abstract public Writable terminate(Writable buffer) throws UDFException;
abstract public void merge(Writable buffer, Writable partial) throws UDFException;
}
```

The most important APIs are iterate, merge, and terminate. The primary logic of UDAFs relies on the implementation of these three APIs. In addition, you must implement a custom writable buffer. As an example, the following figure briefly illustrates the implementation logic and computational flow of the avg (average value) MaxCompute UDAF function.



In the preceding figure, the input data is sliced by a certain size (for description of slicing, see MapReduce). The size of each slice is suitable for a worker to complete in an appropriate period of time. You need to manually configure the size of the slices.

The UDAF calculation process is divided into two phases:

- Phase 1: Each Worker counts the number of data rows and the sum of the data in each slice. The user can regard the counted number and sum as an intermediate result.
- Phase 2: The Worker summarizes the information gained from the previous phase within each slice. In the final output, r.sum / r.count is the average of all input data.

The following example shows how to calculate an average by using a UDAF:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.udf.Aggregator;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.annotation.Resolve;
@Resolve({"double->double"})
public class AggrAvg extends Aggregator {
private static class AvgBuffer implements Writable { private double sum = 0;
private long count = 0;
@Override
public void write(DataOutput out) throws IOException { out.writeDouble(sum);
```

```
out.writeLong(count);
}
@Override
public void readFields(DataInput in) throws IOException { sum = in.readDouble();
count = in.readLong();
}
}
private DoubleWritable ret = new DoubleWritable();
@Override
public Writable newBuffer() { return new AvgBuffer();
}
@Override
public void iterate(Writable buffer, Writable[] args) throws UDFException { DoubleWritable arg = (Doub
leWritable) args[0];
AvgBuffer buf = (AvgBuffer) buffer; if (arg ! = null) {
buf.count += 1; buf.sum += arg.get();
}
}
@Override
public Writable terminate(Writable buffer) throws UDFException { AvgBuffer buf = (AvgBuffer) buffer;
if (buf.count == 0) { ret.set(0);
} else {
ret.set(buf.sum / buf.count);
}
return ret;
}
@Override
public void merge(Writable buffer, Writable partial) throws UDFException { AvgBuffer buf = (AvgBuffer
) buffer;
AvgBuffer p = (AvgBuffer) partial; buf.sum += p.sum;
buf.count += p.count;
}
}
```

Notice

- The SQL syntax used by UDAFs is the same as that used by common built-in aggregate functions. For more information, see Aggregate functions.
- The way to run UDTFs is the same as that to run UDFs. For more information, see Run UDFs.

1.6.8.5. UDTFs

1.6.8.5.1. Overview

Java UDTFs must inherit the com.aliyun.odps.udf.UDTF class. This class requires the implementation of four APIs. The following table lists the definitions of these APIs.

API definitions

API	Description
public void setup(ExecutionContext ctx) throws UDFException	The initialization method to call the user-defined initialization behavior before a UDTF processes the input data. SETUP is called once first in each worker.
public void process(Object[] args) throws UDFException	This method is called by the framework. Each SQL record calls PROCESS once. The parameters of PROCESS are the specified UDTF input parameters in the SQL statement. The input parameters are passed in as Object[], and the results are output by the FORWARD function. You need to call FORWARD in the PROCESS function to determine the output data.
public void close() throws UDFException	The termination method of UDTF. This method is called by the framework for only once after the last record is processed.
public void forward(Object ...o) throws UDFException	You can call the FORWARD method to output data. Each time FORWARD is called, it outputs one record. The record corresponds to the column specified by the UDTF AS clause in the SQL statement.

UDTF example:

```
package org.alidata.odps.udtf.examples;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.UDTFCollector;
import com.aliyun.odps.udf.annotation.Resolve;
import com.aliyun.odps.udf.UDFException;
// TODO define input and output types, e.g., "string,string->string,bigint".
@Resolve({"string,bigint->string,bigint"}) public class MyUDTF extends UDTF {
@Override public void process(Object[] args) throws UDFException { String a = (String) args[0];
Long b = (Long) args[1];
for (String t: a.split("\\s+")) { forward(t, b);
}
}
}
```

The preceding example shows how to create a UDTF in MaxCompute. If this UDTF is named user_udtf, you can run the following SQL statement to call this UDTF:

```
select user_udtf(col0, col1) as (c0, c1) from my_table;
```

The values in my_table col0 and col1 are as follows:

```
+------+------+
| col0 | col1 |
+------+------+
| A B  | 1    |
| C D  | 2    |
+------+------+
```

The result of the SELECT statement is as follows:

```
+----+----+
| c0 | c1 |
+----+----+
| A  | 1  |
| B  | 1  |
| C  | 2  |
| D  | 2  |
+----+----+
```

1.6.8.5.2. UDTF description

Common uses of UDTFs in SQL:

```
select user_udtf(col0, col1) as (c0, c1) from my_table;
select user_udtf(col0, col1) as (c0, c1) from (select * from my_table distribute by col1 sort by col1) t;
```

> **Notice**
>
> The following limits apply to the use of UDTF.
>
> - No other expressions are allowed in a SELECT clause.
>
> ```
> select col0, user_udtf(col0, col1) as (c0, c1) from mytable;
> ```
>
> - UDTFs cannot be nested.
>
> ```
> select user_udtf(mp_udtf(col0,col1)) as (c0,c1)from mytable;
> ```

UDTF examples

The user can use a UDTF to read MaxCompute resources. The following are examples of reading MaxCompute resources by using UDTFs:

1. Write UDTF program. The JAR package (udtfexample1.jar) is exported after compilation.

```
package com.aliyun.odps.examples.udf;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Iterator;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDFException;
import com.aliyun.odps.udf.UDTF;
import com.aliyun.odps.udf.annotation.Resolve;
/**
* project: example_project
* table: wc_in2
* partitions: p2=1,p1=2
* columns: colc,colb
*/
@Resolve({ "string,string->string,bigint,string" }) public class UDTFResource extends UDTF { Execu
tionContext ctx;
long fileResourceLineCount;
long tableResource1RecordCount;
long tableResource2RecordCount;
@Override
public void setup(ExecutionContext ctx) throws UDFException { this.ctx = ctx;
```

```
try {
InputStream in = ctx.readResourceFileAsStream("file_resource.txt");
BufferedReader br = new BufferedReader(new InputStreamReader(in));
String line;
fileResourceLineCount = 0;
while ((line = br.readLine()) ! = null) { fileResourceLineCount++;
}
br.close();
Iterator<Object[]> iterator = ctx.readResourceTable("table_resource1").iterator();
tableResource1RecordCount = 0;
while (iterator.hasNext()) { tableResource1RecordCount++; iterator.next();
}
iterator = ctx.readResourceTable("table_resource2").iterator();
tableResource2RecordCount = 0;
while (iterator.hasNext()) { tableResource2RecordCount++;
iterator.next();
}
} catch (IOException e) { throw new UDFException(e);
}
}
@Override
public void process(Object[] args) throws UDFException { String a = (String) args[0];
long b = args[1] == null ? 0 : ((String) args[1]).length();
forward(a, b, "fileResourceLineCount=" + fileResourceLineCount + "|tableResource1RecordCount="
+ tableResource1RecordCount + "|tableResource2RecordCount=" + tableResource2RecordCount);
}
}
```

2. Add resources to MaxCompute.

```
Add file file_resource.txt;
Add jar udtfexample1.jar;
Add table table_resource1 as table_resource1;
Add table table_resource2 as table_resource2;
```

3. Create UDTF function (mp_udtf) in MaxCompute.

```
create function mp_udtf as com.aliyun.odps.examples.udf.UDTFResource using 'udtfexample1.jar,
file_resource.txt, table_resource1, table_resource2';
```

4. Create resource tables 'table_resource1' and 'table_resource2' in MaxCompute, and insert the corresponding data.
5. Run this UDTF.

```
select mp_udtf("10","20") as (a, b, fileResourceLineCount) from table_resource1;
-- Command output:
+-------+------------+-------+
| a | b     | fileResourceLineCount |
+-------+------------+-------+
| 10 | 2 | fileResourceLineCount=3|tableResource1RecordCount=0|tableResource2RecordCount=0 |
| 10    | 2         | fileResourceLineCount=3|tableResource1RecordCount=0|tableResource2RecordCou
nt=0 |
+-------+------------+-------+
```

Note You can also use the same method to obtain resources. For more information, see MapReduce examples.

UDTF examples — Complex data types

The code in the following example defines a UDF with three overloads. The first overload uses array as the parameter; the second uses map as the parameter; and the third uses struct as the parameter. The third overload uses a struct type as the parameter or returned value, the UDF class must be supplemented with a @Resolve annotation to specify the specific type of struct.

```
@Resolve("struct<a:bigint>,string->string")
public class UdfArray extends UDF {
  public String evaluate(List<String> vals, Long len) {
    return vals.get(len.intValue());
  }
  public String evaluate(Map<String,String> map, String key) {
    return map.get(key);
  }
  public String evaluate(Struct struct, String key) {
    return struct.getFieldValue("a") + key;
  }
}
```

You can import a complex data type in the UDF:

```
create function my_index as 'UdfArray' using 'myjar.jar';
select id, my_index(array('red', 'yellow', 'green'), colorOrdinal) as color_name from colors;
```

1.6.8.6. Python UDFs

1.6.8.6.1. Restricted environment

MaxCompute UDF uses Python V2.7. It executes user codes in a sandbox. The following operations are restricted in the sandbox:

- Read and write local files.
- Start subprocesses.
- Start threads.
- Conduct socket communication.
- Call other systems.

Due to these restrictions, user-uploaded code must all be implemented by Python, as C extension modules are disabled.

In addition, not all modules in the Python standard library are available for use. Modules that involve the preceding features are disabled. Description of available modules in the standard library:

1. All modules implemented purely by Python are available.
2. The following C extension modules are available for use.
   - array
   - audioop
   - binascii
   - _bisect
   - cmath
   - _codecs_cn
   - _codecs_hk
   - _codecs_iso2022
   - _codecs_jp
   - _codecs_kr
   - _codecs_tw
   - _collections
   - cStringIO
   - datetime
   - _functools
   - future_builtins
   - _hashlib
   - _heapq
   - itertools
   - _json
   - _locale
   - _lsprof
   - math
   - _md5
   - _multibytecodec

- operator
- _random
- _sha256
- _sha512
- _sha
- _struct
- strop
- time
- unicodedata
- _weakref
- cPickle

3. Some modules have limited functionality. For example, the sandbox limits the size that user codes can write to the standard output and standard error output. `sys.stdout and sys.stderr` can write up to 20 KB. Any remaining characters are ignored.

1.6.8.6.2. Third-party libraries

Common third-party libraries are installed in the operating environment to supplement the standard library. The supported third-party libraries include NumPy.

> **Warning** The use of third-party libraries is also subject to restrictions. For example, local or remote I/O operations are prohibited. Therefore, the related APIs in the third-party libraries are disabled.

1.6.8.6.3. Types of parameters and returned values

You can run the following command to specify the types of parameters and returned values:

```
@odps.udf.annotate(signature)
```

Python UDFs support the following MaxCompute SQL data types: bigint, string, double, boolean, and datetime. Before you run a SQL statement, you must specify the parameter types and returned value types of all functions. Python is a dynamically-typed language. You need to add decorators to the UDF class to specify the function signature.

The function signature is specified by a string. The syntax is as follows:

```
arg_type_list '->' type_list
arg_type_list: type_list | '*' | ''
type_list: [type_list ','] type
type: 'bigint' | 'string' | 'double' | 'boolean' | 'datetime'
```

Note

- The part to the left of the arrow indicates the type of parameter. The part to the right of the arrow indicates the type of returned value.
- The returned value of a UDTF can contain multiple columns. The returned value of a UDF or UDAF can contain only one column.
- * represents a variable argument. If a variable argument is specified, the UDF, UDTF, or UDAF can match any type of parameter.

Examples of valid signature:

```
'bigint,double->string'
-- The parameter is of the bigint or double type, and the returned value is of the string type.
'bigint,boolean->string,datetime'
-- The UDTF parameter is of the bigint or boolean type, and the returned value is of the string or datet
ime type.
'*->string'
-- Specify a variable argument: The input parameter can be of any type, and the returned value is of t
he string type.
'->double'
-- The parameter is NULL and the returned value is of the double type.
```

If an invalid signature is found during query parsing, an error is returned and the execution is banned. During execution, the UDF parameter with the type specified by the function signature is transferred to the user. The user returned value must be of the type specified by the function signature. Otherwise, an error is returned. The following table shows the mappings between MaxCompute SQL types and Python types.

Mapping

MaxCompute SQL type	Python type
Bigint	int
String	str
Double	float
Boolean	bool
Datetime	int

> **Note**
>
> - A value of the datetime type is passed to user code as the int type. The value is the number of milliseconds that have elapsed since the epoch time. You can use the datetime module in the Python standard library to process the datetime type.
> - NULL corresponds to none in Python.

In addition, the parameter of odps.udf.int(value[, silent=True]) is modified. Parameter silent is added. If silent is true and the value cannot be converted to the int type, none is returned instead of an error.

1.6.8.6.4. UDFs

Implementing a Python UDF is as easy as defining a new-style class and implementing the evaluate method.

Example:

```
from odps.udf import annotate
@annotate("bigint,bigint->bigint")
class myplus (object ):
  def evaluate (self, arg0, arg1 ):
    If none in (arg0, arg1 ):
      return none
    return arg0 + arg1
```

> **Notice** A Python UDF must have its signature specified through annotate.

1.6.8.6.5. UDAFs

Description:

- class odps.udf.BaseUDAF: inherit this class to implement a Python UDAF.
- BaseUDAF.new_buffer(): implement this method and return the median 'buffer' of the aggregate function. Buffer must be mutable object (such as list and dict). The size of the buffer should not increase with the amount of data. The buffer size should not exceed 2 MB after marshal.
- BaseUDAF.iterate(buffer[, args, ...]): This method aggregates args into the median buffer.
- BaseUDAF.merge(buffer, pbuffer): This method aggregates two median buffers; that is, aggregate pbuffer into buffer.
- BaseUDAF.terminate(buffer): This method converts the median 'buffer' into the MaxCompute SQL basic types.

The following example shows how to calculate an average by using a UDAF:

```
#coding:utf-8
from odps.udf import annotate
from odps.udf import BaseUDAF
@annotate('double->double')

class Average(BaseUDAF):
   def new_buffer(self):
      return [0, 0]
   def iterate(self, buffer, number):
      If number is not None:
         buffer[0] += number
         buffer[1] += 1
   def merge(self, buffer, pbuffer):
      buffer [0] + = pbuffer [0]
      buffer [1] + = pbuffer [1]
   def terminate (self, buffer ):
      If buffer [1] = 0:
         return 0.0
      return buffer[0] / buffer[1]
```

1.6.8.6.6. UDTFs

The parameters are described as follows.

Parameters

Parameter	Description
class odps.udf.BaseUDTF	Base class for a Python UDTF. Users inherit this class and implement methods such as PROCESS and CLOSE.
BaseUDTF.init()	Initialization method. To implement this method for an inherited class, you must call the initialization method super(BaseUDTF, self).init() for the base class at the beginning . The INIT method will only be called once during the entire UDTF life cycle; that is, before the first record is processed. When the UDTF needs to save internal states, all states can be initialized in this method.
BaseUDTF.process([args, ...])	The method is called by the MaxCompute SQL framework. The process method is called for each record passed in from SQL. The parameters passed into the process method are the parameters passed into the UDTF in SQL statements.

Parameter	Description
BaseUDTF.forward([args, ...])	The UDTF output method, which is called by user code. Each time FORWARD is called, one record is output. The parameters of FORWARD are the UDTF output parameters specified in SQL statements.
BaseUDTF.close()	The UDTF termination method. This method is called by the MaxCompute SQL framework.This method is called only once, after the last record is processed.

Example:

```
#coding:utf-8
# explode. py
from odps.udf import annotate

from odps.udf import BaseUDTF
@annotate('string -> string')
class Explode(BaseUDTF):
-- Output string as multiple comma-separated records.
  def process(self, arg):
     props = arg.split(',')
      for p in props:
        self.forward(p)
```

Notice A Python UDTF can also specify the parameter type or returned value type without adding 'annotate'. In this case, the function can match any input parameter in SQL. The type of returned value cannot be deduced, but all output parameters will be considered to be of the string type. Therefore, when FORWARD is called, all output values must be converted into values of the string type.

1.6.8.6.7. Reference resources

You can reference file and table resources in Python UDF through the odps.distcache module.

Syntax for referencing file resources:

```
odps.distcache.get_cache_file(resource_name)
```

Note

- Description: returns the content of the specified resource. resource_name is a string that corresponds to the name of an existing resource in the current project. If the resource name is invalid or does not exist, an error is returned.
- Returned value: returns file-like object. After this object is used, the caller must call the CLOSE method to release the resource file that is opened.

Example:

```
from odps.udf import annotate
from odps.distcache import get_cache_file
@annotate('bigint->string')
class DistCacheExample(object):
def __init__(self):
    cache_file = get_cache_file('test_distcache.txt')
    kv = {}
    for line in cache_file:
        line = line.strip()
        if not line:
            continue
        k, v = line.split()
        kv[int(k)] = v
    cache_file.close()
    self.kv = kv
def evaluate(self, arg):
    return self.kv.get(arg)
```

Command syntax:

```
odps.distcache.get_cache_table(resource_name)
```

Note

- Description: returns the content of the specified resource table. resource_name is a string that corresponds to the name of an existing resource table in the current project. If the resource table name is invalid or does not exist, an error is returned.
- Returned value: returns a value of the generator type. The caller traverses the table to obtain the content. Each time the caller traverses the table, a record is obtained in the form of a tuple.

Example:

```
from odps.udf import annotate
from odps.distcache import get_cache_table
@annotate('->string')
class DistCacheTableExample(object):
    def __init__(self):
        self.records = list(get_cache_table('udf_test'))
        self.counter = 0
        self.ln = len(self.records)
    def evaluate(self):
        if self.counter > self.ln - 1:
            return None
        ret = self.records[self.counter]
        self.counter += 1
        return str(ret)
```

1.6.9. UDTs

1.6.9.1. Overview

User-defined types (UDTs) are introduced in MaxCompute 2.0 for the latest version of the SQL engine. UDTs allow you to reference classes or objects of third-party languages in SQL statements to obtain data or call methods.

UDTs are typically applied in the following scenarios:

- **Scenario 1:** MaxCompute does not have built-in functions to complete tasks that can be easily performed using other languages. For example, there are some tasks that can be performed by calling a single built-in Java class. Performing these tasks with user defined functions (UDFs) is complex.
- **Scenario 2:** You need to call a third-party library in SQL statements to implement the corresponding feature. You want to use a feature provided by a third-party library directly in a SQL statement, instead of wrapping the feature inside a UDF.
- **Scenario 3:** SELECT TRANSFORM allows you to include objects and classes in SQL statements to make these SQL statements easier to read and maintain. For some languages, such as Java, the source code can be only executed after it is compiled. You want to reference objects and classes of these languages in SQL statements.

> **Notice**
>
> - UDTs only support Java.
> - All operators use the semantics of MaxCompute SQL.
> - UDTs cannot be used as shuffle keys in the JOIN, GROUP BY, DISTRIBUTE BY, SORT BY, ORDER BY, and CLUSTER BY clauses.
> - DDL statements do not support UDTs. You cannot create tables that contain UDT objects. The final output cannot be UDT types.

1.6.9.2. Feature summary

UDTs allow you to reference classes or objects of third-party languages in SQL statements to obtain data or call methods.

The UDTs supported in MaxCompute are very different from those in other SQL engines.

UDTs supported by other SQL engines are similar to the struct composite type in MaxCompute. UDTs supported by MaxCompute are similar to the CREATE TYPE statement. A UDT contains both fields and methods. Additionally, MaxCompute does not require that you use Data Definition Language (DDL) statements to define type mappings. MaxCompute allows you to reference types directly in SQL statements.

Example:

```
set odps.sql.type.system.odps2=true;
SELECT Integer.MAX_VALUE;
-- A similar output is displayed:
+------------+
| max_value |
+------------+
| 2147483647 |
+------------+
```

The expression in the preceding SELECT statement is similar to a Java expression and executed in the same manner as it would in Java. The expression specifies a UDT in MaxCompute.

You can use UDFs to implement all features provided by UDTs, but with some complexity. If you use a UDF to implement the same feature, you need to follow these steps:

1. Define a UDF class.

   ```
   package com.aliyun.odps.test;
   public class IntegerMaxValue extends com.aliyun.odps.udf.UDF {
   public Integer evaluate() {
    return Integer.MAX_VALUE;
   }
   }
   ```

2. Compile the UDF as a JAR package. Upload the JAR package and create a function.

   ```
   add jar odps-test.jar;
   create function integer_max_value as 'com.aliyun.odps.test.IntegerMaxValue' using 'odps-test.jar'
   ;
   ```

3. Call the function in a SQL statement.

   ```
   select integer_max_value();
   ```

A UDT simplifies this procedure. By using UDTs, you can use features provided by other

languages in SQL statements.

1.6.9.3. Feature description

The example described in the Feature overview topic demonstrates how to use user-defined types (UDTs) to access the static fields of Java classes. UDTs can be used to implement a number of functions. The following example shows a UDT execution procedure and its features.

```
-- Sample data
@table1 := select * from values ('10000000000000000000') as t(x);
@table2 := select * from values (100L) as t(y);
-- Code logic
@a := select new java.math.BigInteger(x) x from @table1;        -- Create an object by using the new m
ethod.
@b := select java.math.BigInteger.valueOf(y) y from @table2;     -- Call a static method.
select /*+mapjoin(b)*/ x.add(y).toString() from @a a join @b b;   -- Call an instance method.
```

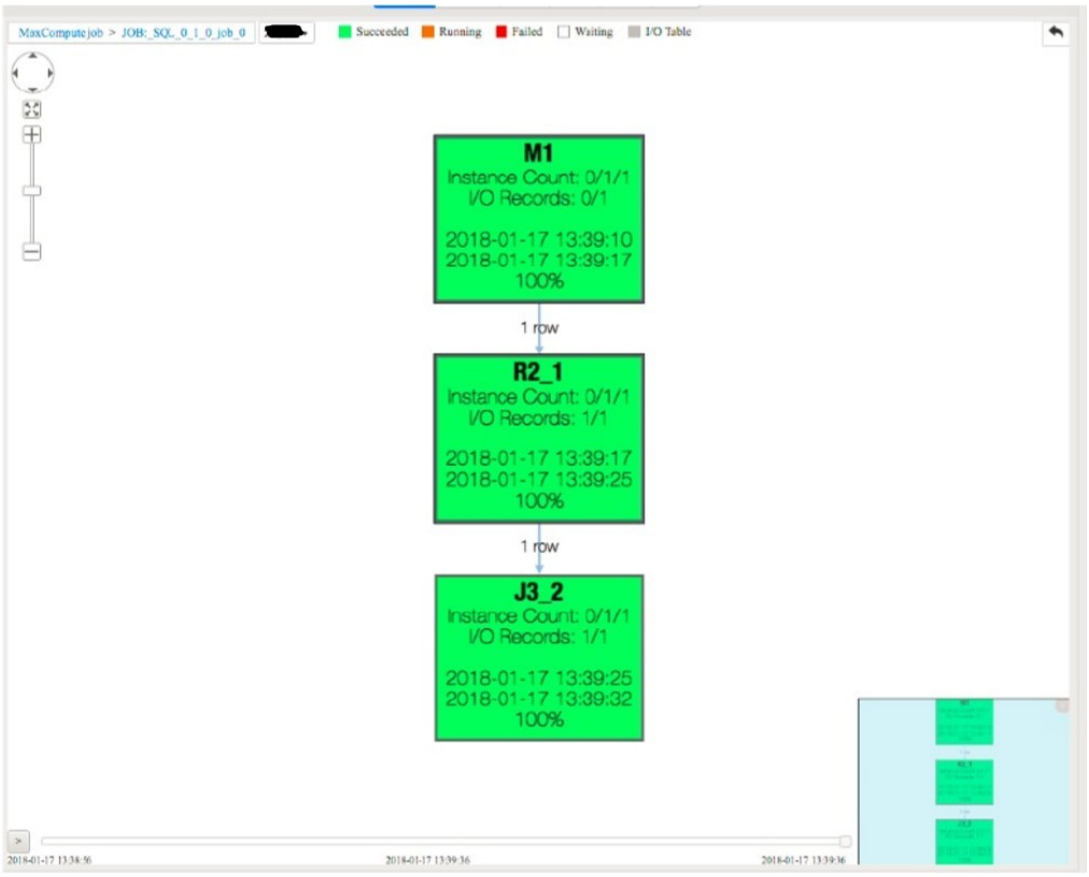
The following result is returned:

```
100000000000000000100
```

> **Note** This example also shows how to use subqueries with UDT columns. User-defined functions (UDFs) cannot be used in such subqueries. Variable a in the x column is of the java.math.BigInteger class, not a built-in class. You can pass UDT data to another operator and then call the required method. You can also use UDT data in data shuffling.

UDT execution

Example



The preceding figure shows the three stages of a UDT: M1, R2, and J3. Only the *new java.math.BigInteger(x)* method is called at the M1 stage. The *java.math.BigInteger.valueOf(y)* and *x.add(y).toString()* methods are called at the J3 stage.

If a JOIN clause is used in MapReduce, data must be reshuffled. As a result, data is processed at multiple stages. Data is processed at different stages or even by different processes or physical machines. The UDT encapsulates these stages and functions as a JVM.

Description

- UDTs support only Java.
- UDTs also allow you to upload JAR packages and directly reference these packages. Some flags are provided for UDTs.

- **set odps.sql.session.resources**: specifies the resource that you want to reference. Separate multiple resources with commas (,). For example, you can set this flag to `foo.sh,bar.txt`. Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.resources=odps-test.jar; -- Specify the JAR package that you want to refere
nce. Before you reference the JAR package, upload the package to your project.
select new com.aliyun.odps.test.IntegerMaxValue().evaluate();
```

> **Notice** This flag is the same as the flag that is used to specify resources in the SELECT TRANSFORM statement. Therefore, this flag affects JAR package uploading in UDTs and resource settings in the SELECT TRANSFORM statement.

- **odps.sql.session.java.imports**: specifies the default Java package. Separate multiple Java packages with commas (,). This flag is similar to the IMPORT statement in Java. You can specify a classpath, such as `java.math.BigInteger`, or use `*`. `Static import` is not supported. Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.resources=odps-test.jar;
set odps.sql.session.java.imports=com.aliyun.odps.test. *;  -- Specify the default Java package.
select new IntegerMaxValue().evaluate();
```

- UDTs allow you to:
  - Create objects by using the new method.
  - Create arrays by using the new method, including ArrayList initialization. Example: `new Integer[] { 1, 2, 3 }`.
  - Call methods, including static methods. You can create objects in the factory method pattern.
  - Access fields, including static fields.

> **Notice**
> - Identifiers in UDTs contain package names, class names, method names, and field names. All identifiers are case-sensitive.
> - UDTs support SQL type conversions, such as cast(1 as java.lang.Object). UDTs do not support Java type conversions, such as (Object)1.
> - Anonymous classes and lambda expressions are not supported.
> - Functions that do not return values cannot be called in UDTs.
>
> > **Note** UDTs are used in expressions. Functions that do not return values cannot be called in expressions.

- All SDK for Java classes can be referenced by UDTs. The JDK runtime environment is JDK 1.8. Later versions may not be supported.

- All operators use the semantic of MaxCompute SQL. The result of `String.valueOf(1) + String.valueOf(2)` is 3. The two strings are implicitly converted to DOUBLE-type values and summed. If you use Java string concatenation to combine the strings, the result is 12.

  You may be confused about the role of the `=` operator. The `=` operator in SQL statements is used as a comparison operator. It is used to compare one expression with another expression. You must call the equals method in Java to compare whether two objects are equivalent. The = operator cannot be used to verify the equivalence of two objects.

- Java data types are mapped to built-in data types. The mapping can be applied to UDTs.
  - You can directly call the method of the Java type to which the built-in type is mapped. Example: `'123'.length()` , `1L.hashCode()` .
  - UDTs can be used in built-in functions and UDFs. For example, in `chr(Long.valueOf('100'))` , `Long.valueOf` returns a value of the `java.lang.Long` type. The CHR built-in function supports the built-in BIGINT type.
  - The data of a Java primitive type is automatically converted to the boxing type and the preceding two rules are applied.

  > **Notice** For some new built-in data types, you must use `set odps.sql.type.system.odps2=true;` to declare these types. Otherwise, an error occurs.

- UDTs completely support Java generics. For example, based on the parameter type, the compiler can determine that the value returned by `java.util.Arrays.asList(new java.math.BigInteger('1'))` is `java.util.List<java.math.BigInteger>` .

  > **Notice** You must specify the type parameter in a constructor function or use java.lang.Object. This is the same as Java. For example, the result of `new java.util.ArrayList(java.util.Arrays.asList('1', '2'))` is of the `java.util.ArrayList<Object>` type. The result of `new java.util.ArrayList<String>(java.util.Arrays.asList('1', '2'))` is of the `java.util.ArrayList<String>` type.

- UDTs do not have a clear definition of **object equality**. This is caused by data reshuffling. The JOIN example shows that objects may be transmitted between different processes or physical machines. During transmission, an object may be referenced as two different objects. For example, an object may be shuffled to two machines and then reshuffled.

  Therefore, when you use UDTs, you must use the equals method instead of the = operator to equate two objects.

  > **Note** Objects in the same row or column are correlated in some way. However, a correlation between objects in different rows or columns cannot be ensured.

- UDTs cannot be used as shuffle keys in clauses, such as JOIN, GROUP BY, DISTRIBUTE BY, SORT BY, ORDER BY, or CLUSTER BY.

UDTs can be used at the stages in expressions, but cannot be used as outputs. For example, you cannot call the `group by new java.math.BigInteger('123')` method. However, you can call the `group by new java.math.BigInteger('123').hashCode()` method. This is because the value returned by hashCode is an int.class type, which can be used as the built-in INT type.

- The following type conversion rules are extended in UDTs:
  - UDT objects can be implicitly converted to the objects of their base classes.
  - UDT objects can be forcibly converted to the objects of their base classes or subclasses.
  - The data type conversion for two objects without inheritance follows native conversion rules.

    > **Notice** The conversion may cause data changes. For example, data of the java.lang.Long type can be forcibly converted to the java.lang.Integer type. This conversion uses the rules that are used to convert the built-in BIGINT type to the INT type. This process may cause data changes or even data precision loss.

- UDT objects cannot be saved or added to tables. DDL statements do not support UDTs. You cannot create tables that contain UDT objects unless the data type is implicitly converted to one of the built-in types. In addition, the output cannot be a UDT. However, you can call the toString() method to convert the data type to the java.lang.String type because the toString() method supports all Java classes. You can use this method to check UDT data during debugging.

  You can also add the `set odps.sql.udt.display.tostring=true;` flag to enable MaxCompute to convert all output UDT data to strings by calling the `java.util.Objects.toString(...)` method for debugging.

  > **Note** This flag is typically used for debugging because it can be applied only to PRINT statements. It cannot be applied to INSERT statements.

  BINARY is a built-in type and supports automatic serialization. You can save the byte[] arrays. The saved byte[] arrays can be deserialized to the BINARY type.

  Some classes may have their own serialization and deserialization methods, such as protobuffer. To save UDTs, you must call serialization and deserialization methods to convert the data type to BINARY.

- You can use UDTs to achieve the feature provided by the SCALAR function. You can use the COLLECT_LIST and EXPLODE built-in functions with UDTs to achieve the features provided by aggregate and table-valued functions.
- UDTs support resource access. You can call the `com.aliyun.odps.udf.impl.UDTExecutionContext.get()` static method to obtain the ExecutionContext object. Then, use the object to access the current execution context and then to access resources, such as files and tables.

1.6.9.4. More examples

1.6.9.4.1. Example of using Java arrays

Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.udt.display.tostring=true;
select
   new Integer[10],    -- Create an array that contains 10 elements.
   new Integer[] {c1, c2, c3},  -- Create an array that contains three elements by initializing an ArrayList.
   new Integer[][] { new Integer[] {c1, c2}, new Integer[] {c3, c4} },  -- Create a multidimensional array.
   new Integer[] {c1, c2, c3} [2], -- Access the elements in the array using indexes.
   java.util.Arrays.asList(c1, c2, c3);    -- This is another way to create a built-in array. It creates a List<I
nteger>, which can be used as an array<int>.
from values (1,2,3,4) as t(c1, c2, c3, c4);
```

1.6.9.4.2. Example of using JSON

The runtime of UDT carries a GSON dependency (version 2.2.4), which can be directly used in GSON.

Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*,java,com.google.gson. *; -- To import multiple packages, s
eparate the packages with commas (,).
@a := select new Gson() gson;   -- Create a GSON object.
select
gson.toJson(new ArrayList<Integer>(Arrays.asList(1, 2, 3))), -- Convert an object to a JSON string.
cast(gson.fromJson('["a","b","c"]', List.class) as List<String>) --Deserialize the JSON string. GSON also f
orcibly converts the deserialized result from List<Object> type to List<String> type.
from @a;
```

Compared with built-in function **GET_JSON_OBJECT**, this method is simple and improves efficiency by extracting content from the JSON string and deserializing the string to a supported data type.

In addition to GSON dependencies, MaxCompute runtime also carries other dependencies, including commons-logging (1.1.1), commons-lang (2.5), commons-io (2.4), and protobuf-java (2.4.1).

1.6.9.4.3. Example of using composite types

Built-in types of array and map are mapped to java.util.List and java.util.Map, respectively.

- Java objects in classes calling the java.util.List or java.util.Map API can be used in MaxCompute SQL composite type data processing.
- Array and map type data in MaxCompute can directly call the java.util.List or java.util.Map API.

Example:

```
set odps.sql.type.system.odps2=true;
set odps.sql.session.java.imports=java.util.*;
select
   size(new ArrayList<Integer>()),       -- Call built-in function size to obtain the size of the ArrayList.
   array(1,2,3).size(),                  -- Call the List method for built-in type array.
   sort_array(new ArrayList<Integer>()),  -- Sort the data in the ArrayList.
   al[1],                                -- The Java List method does not support indexing. However, the array type s
upports indexing.
   Objects.toString(a),       -- With this method, you can convert array type to string type data.
   array(1,2,3).subList(1, 2)            -- Get a sublist.
from (select new ArrayList<Integer>(array(1,2,3)) as al, array(1,2,3) as a) t;
```

1.6.9.4.4. Example of aggregation

To achieve aggregation with UDTs, you must first use built-in function COLLECT_SET or COLLECT_LIST to convert the data to the List type and then call the UDT methods to aggregate the data.

The following example shows how to obtain the median from BigInteger data. You cannot directly call the built-in MEDIAN function because the data is java.math.BigInteger type.

```
set odps.sql.session.java.imports=java.math.*;
@test_data := select * from values (1),(2),(3),(5) as t(value);
@a := select collect_list(new BigInteger(value)) values from @test_data;  -- Aggregate the data to a li
st.
@b := select sort_array(values) as values, values.size() cnt from @a;  -- To obtain the median, first sor
t the data.
@c := select if(cnt % 2 == 1, new BigDecimal(values[cnt div 2]), new BigDecimal(values[cnt div 2 - 1].add
(values[cnt div 2])).divide(new BigDecimal(2))) med from @b;
-- Final output.
select med.toString() from @c;
```

You cannot use the COLLECT_LIST function to implement partial aggregation because it aggregates all data. It is more efficient to use the built-in aggregator or UDAF object. We recommend that you use the built-in aggregator. Aggregating all data in a group increases the risk of data skew.

If the logic of the UDAF object is to aggregate all data in a similar manner to built-in function WM_CONCAT, using the COLLECT_LIST function is more efficient than using the UDAF object.

1.6.9.4.5. Example of using table-valued functions

Table-valued functions allow you to input and output multiple rows and columns. To input or output multiple rows and columns, follow these steps:

1. For more information about how to input multiple rows or columns, see the example of using

aggregate functions.

2. To output multiple rows, you can use a UDT to define a Collection type (List or Map), and then call the EXPLODE function to split the collection into multiple rows.
3. A UDT can contain multiple fields. You can retrieve the data from the fields by calling different getter methods. The data is then output in multiple rows.

The following example shows how to split a JSON string and output the result as multiple columns:

```
@a := select '[{"a":"1","b":"2"},{"a":"1","b":"2"}]' str; -- Sample data
@b := select new com.google.gson.Gson().fromJson(str, java.util.List.class) l from @a; -- Deserialize th
e JSON string.
@c := select cast(e as java.util.Map<Object,Object>) m from @b lateral view explode(l) t as e;  -- Call t
he EXPLODE function to split the string.
@d := select m.get('a') as a, m.get('b') as b from @c; -- Output the splitting result in multiple columns.
select a.toString() a, b.toString() b from @d; -- The final output. Columns a and b in variable d are of t
he Object type.
```

1.6.9.5. Feature advantages

UDT has the following features:

- Easy to use. You do not need to define any functions.
- To improve the flexibility of SQL, all JDK supported features can be used directly.
- You can directly reference objects and classes of other languages in SQL statements.
- You can directly reference the libraries of other language and reuse code that you have written in other languages.
- You can create object-oriented features.

1.6.9.6. Performance advantages

UDTs and UDFs use similar execution procedures and provide similar performance. However, UDTs have higher performance in certain scenarios where the compute engine has been greatly improved.

- Deserialization is not required for objects in only one process. Deserialization is required only when the objects are transmitted among processes. This means that UDT do not incur any serialization or deserialization overhead when no data reshuffling is performed, such as calling the join or aggregator function.
- UDTs suffer no performance loss from reflection because the runtime of UDTs is based on Codegen, rather than based on reflection.
- Multiple UDTs can be wrapped into a single function call and executed together. In the following example, a single UDT is being called. UDTs focus on small-granularity data processing. This does not incur additional overhead for the API where multiple functions are called.

  ```
  values[x].add(values[y]).divide(java.math.BigInteger.valueOf(2))
  ```

1.6.9.7. Security advantages

UDTs are restricted in the Java sandbox model similar to UDFs. To perform restricted operations, you must enable sandbox isolation or apply to join the sandbox whitelist.

1.6.10. UDJ

1.6.10.1. Overview

MaxCompute provides multiple JOIN methods natively, including INNER JOIN, RIGHT JOIN, OUTER JOIN, LEFT JOIN, FULL JOIN, SEMIJOIN, and ANTISEMIJOIN methods. You can use these native JOIN methods in most scenarios. However, these methods cannot handle multiple tables.

In most cases, you can build your code framework using UDFs. However, the current UDF, UDTF, and UDAF frameworks only can handle one table at a time. To perform user-defined operations for multiple tables, you have to use native JOIN methods, UDFs, UDTFs, and complex SQL statements. In certain cases when you handle multiple tables, you must use a custom MapReduce framework instead of SQL to complete the required task.

In any situation, these operations require technological expertise and may cause the following problems:

- Calling multiple JOIN methods in SQL statements can lead to computational black box that is complex and difficult to execute with minimal overheads.
- Using MapReduce even make optimal execution of code becomes impossible. Most of the MapReduce code is written in Java. The execution of the MapReduce code is less efficient than the execution of MaxCompute code generated by the LLVM code generator at an optimized native runtime.

With the addition of the MaxCompute 2.0 compute engine, the user defined join (UDJ) API has been added to the user defined function (UDF) framework. This API allows you to handle multiple tables and simplifies operations performed in the underlying MapReduce distributed system.

1.6.10.2. UDJ usage

1.6.10.2.1. Examples

The following example describes how to use UDJ in MaxCompute.

This example uses the payment table and the user_client_log table.

- The payment (user_id string,time datetime,pay_info string) table stores the payment information of a user. Each payment record includes the user ID, payment time, and the payment details.
- The user_client_log (user_id string,time datetime,content string) table stores user client records, including the user ID, operation time, and operation.

Requirements: For each record in the user_client_log table, locate the payment record that has the time closest to the operation time, and join and output the content of both records.

To complete this task by using standard join methods, you would need to join the two tables based on their common user_id fields, and then locate the payment record and operation that most closely match each other's time. The SQL statement may be written as follows:

```
SELECT
  p.user_id,
  p.time,
  merge(p.pay_info, u.content)
FROM
  payment p RIGHT OUTER JOIN user_client_log u
ON p.user_id = u.user_id and abs(p.time - u.time) = min(abs(p.time - u.time))
```

However, when you join two rows in the tables, you must calculate the minimum difference between the p.time and u.time under the same user_id, and the aggregate function cannot be called in the join condition. Because of this, this task cannot be completed by calling the standard JOIN method.

Can we use UDJ to solve this problem? Yes. The following topics describe how to use UDJ to satisfy the preceding requirements.

1.6.10.2.2. Use Java to write the UDJ code

Prerequisites

UDJ is a new feature, so a new SDK is required.

```
<dependency>
  <groupId>com.aliyun.odps</groupId>
  <artifactId>odps-sdk-udf</artifactId>
  <version>0.30.0</version>
  <scope>provided</scope>
</dependency>
```

The SDK contains a new abstract class UDJ. All UDJ features can be implemented through this class.

Sample code

The following sample code is used for reference only.

```
package com.aliyun.odps.udf.example.udj;
import com.aliyun.odps.Column;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.Yieldable;
import com.aliyun.odps.data.ArrayRecord;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.udf.DataAttributes;
import com.aliyun.odps.udf.ExecutionContext;
import com.aliyun.odps.udf.UDJ;
import com.aliyun.odps.udf.annotation.Resolve;
```

```
import com.aliyun.odps.udf.annotation.Resolve;
import java.util.ArrayList;
import java.util.Iterator;
/** For each record of right table, find the nearest record of left table and
 * merge two records.
 */
@Resolve("->string,bigint,string")
public class PayUserLogMergeJoin extends UDJ {
  private Record outputRecord;
  /** Will be called prior to the data processing phase. User could implement
   * this method to do initialization work.
   */
  @Override
  public void setup(ExecutionContext executionContext, DataAttributes dataAttributes) {
    //
    outputRecord = new ArrayRecord(new Column[]{
      new Column("user_id", OdpsType.STRING),
      new Column("time", OdpsType.BIGINT),
      new Column("content", OdpsType.STRING)
    });
  }
  /** Override this method to implement join logic.
   * @param key Current join key
   * @param left Group of records of left table corresponding to the current key
   * @param right Group of records of right table corresponding to the current key
   * @param output Used to output the result of UDJ
   */
  @Override
  public void join(Record key, Iterator<Record> left, Iterator<Record> right, Yieldable<Record> output)
{
    outputRecord.setString(0, key.getString(0));
    if (! right.hasNext()) {
      // Empty right group, do nothing.
      return;
    } else if (! left.hasNext()) {
      // Empty left group. Output all records of right group without merge.
      while (right.hasNext()) {
        Record logRecord = right.next();
        outputRecord.setBigint(1, logRecord.getDatetime(0).getTime());
        outputRecord.setString(2, logRecord.getString(1));
        output.yield(outputRecord);
```

```
    }
    return;
  }
  ArrayList<Record> pays = new ArrayList<>();
  // The left group of records will be iterated from the start to the end
  // for each record of right group, but the iterator cannot be reset.
  // So we save every records of left to an ArrayList.
  left.forEachRemaining(pay -> pays.add(pay.clone()));
  while (right.hasNext()) {
    Record log = right.next();
    long logTime = log.getDatetime(0).getTime();
    long minDelta = Long.MAX_VALUE;
    Record nearestPay = null;
    // Iterate through all records of left, and find the pay record that has
    // the minimal difference in terms of time.
    for (Record pay: pays) {
      long delta = Math.abs(logTime - pay.getDatetime(0).getTime());
      if (delta < minDelta) {
        minDelta = delta;
        nearestPay = pay;
      }
    }
    // Merge the log record with nearest pay record and output to the result.
    outputRecord.setBigint(1, log.getDatetime(0).getTime());
    outputRecord.setString(2, mergeLog(nearestPay.getString(1), log.getString(1)));
    output.yield(outputRecord);
  }
 }
 String mergeLog(String payInfo, String logContent) {
  return logContent + ", pay " + payInfo;
 }
 @Override
 public void close() {
 }
}
```

Notice In this example, the NULL values in the entries are not processed. To simplify the data processing procedure, assume that no NULL values are contained in the tables.

Each time you call this JOIN method of UDJ, records that match the same key in the two tables are returned. Therefore, UDJ searches all records in the payment table to locate the record with the time closest to each record in the user_client_log table.

Assume that the user only has a few payment records. In this case, you can load the data in the payment table to the memory. Typically, there is sufficient memory to store the user payment data generated each day. What if this assumption is invalid? How can we resolve this issue? This issue will be discussed in **Pre-sorting**.

1.6.10.2.3. Create a UDJ function in MaxCompute

After you have written the UDJ code in Java, upload the code to MaxCompute SQL as a plug-in. You must have registered the code with MaxCompute first.

Assume that the code is compressed into JAR package odps-udj-example.jar. Use the Add JAR command to upload the JAR package to MaxCompute.

```
add jar odps-udj-example.jar;
```

Execute the CREATE FUNCTION statement to create UDJ function *pay_user_log_merge_join*, using JAR package odps-udj-example.jar and Java class *com.aliyun.odps.udf.example.udj.PayUserLogMergeJoin*.

```
create function pay_user_log_merge_join
 as 'com.aliyun.odps.udf.example.udj.PayUserLogMergeJoin'
 using 'odps-udj-example.jar';
```

1.6.10.2.4. Use UDJ in MaxCompute SQL

After you have registered UDJ in the database, UDJ can be used in MaxCompute SQL.

1. Create a sample source table.

   ```
   create table payment (user_id string,time datetime,pay_info string);
   create table user_client_log(user_id string,time datetime,content string);
   ```

2. Create sample data.

   > **Notice** The data in this example is only used for reference. You may need to create different data in actual operations.

```
 -- Create data in the payment table
INSERT OVERWRITE TABLE payment VALUES
('1335656', datetime '2018-02-13 19:54:00', 'PEqMSHyktn'),
('2656199', datetime '2018-02-13 12:21:00', 'pYvotuLDIT'),
('2656199', datetime '2018-02-13 20:50:00', 'PEqMSHyktn'),
('2656199', datetime '2018-02-13 22:30:00', 'gZhvdySOQb'),
('8881237', datetime '2018-02-13 08:30:00', 'pYvotuLDIT'),
('8881237', datetime '2018-02-13 10:32:00', 'KBuMzRpsko'),
('9890100', datetime '2018-02-13 16:01:00', 'gZhvdySOQb'),
('9890100', datetime '2018-02-13 16:26:00', 'MxONdLckwa')
;
-- Create data in the user_client_log table
INSERT OVERWRITE TABLE user_client_log VALUES
('1000235', datetime '2018-02-13 00:25:36', 'click FNOXAibRjkIaQPB'),
('1000235', datetime '2018-02-13 22:30:00', 'click GczrYaxvkiPultZ'),
('1335656', datetime '2018-02-13 18:30:00', 'click MxONdLckpAFUHRS'),
('1335656', datetime '2018-02-13 19:54:00', 'click mKRPGOciFDyzTgM'),
('2656199', datetime '2018-02-13 08:30:00', 'click CZwafHsbJOPNitL'),
('2656199', datetime '2018-02-13 09:14:00', 'click nYHJqIpjevkKToy'),
('2656199', datetime '2018-02-13 21:05:00', 'click gbAfPCwrGXvEjpI'),
('2656199', datetime '2018-02-13 21:08:00', 'click dhpZyWMuGjBOTJP'),
('2656199', datetime '2018-02-13 22:29:00', 'click bAsxnUdDhvfqaBr'),
('2656199', datetime '2018-02-13 22:30:00', 'click XIhZdLaOocQRmrY'),
('4356142', datetime '2018-02-13 18:30:00', 'click DYqShmGbIoWKier'),
('4356142', datetime '2018-02-13 19:54:00', 'click DYqShmGbIoWKier'),
('8881237', datetime '2018-02-13 00:30:00', 'click MpkvilgWSmhUuPn'),
('8881237', datetime '2018-02-13 06:14:00', 'click OkTYNUHMqZzlDyL'),
('8881237', datetime '2018-02-13 10:30:00', 'click OkTYNUHMqZzlDyL'),
('9890100', datetime '2018-02-13 16:01:00', 'click vOTQfBFjcgXisYU'),
('9890100', datetime '2018-02-13 16:20:00', 'click WxaLgOCcVEvhiFJ')
;
```

3. In MaxCompute SQL, use the UDJ function you have created:

```
SELECT r.user_id, from_unixtime(time/1000) as time, content FROM (
SELECT user_id, time as time, pay_info FROM payment
) p JOIN (
SELECT user_id, time as time, content FROM user_client_log
) u
ON p.user_id = u.user_id
USING pay_user_log_merge_join(p.time, p.pay_info, u.time, u.content)
r
AS (user_id, time, content)
;
```

> **Note** The syntax of UDJ is similar to that of the standard JOIN statement. The only difference is that the USING clause is added to UDJ.

Description:

- **pay_user_log_merge_join** is the name of the UDJ function in SQL.
- **(p.time, p.pay_info, u.time, u.content)** are the columns used in these two tables.
- **r** is the alias of the result returned by the UDJ function. You can reference this alias in other SQL statements.
- **(user_id, time, content)** are the columns returned by the UDJ function.

4. Execute this SQL statement. A similar output is displayed:

```
+---------+------------+---------+
| user_id | time       | content |
+---------+------------+---------+
| 1000235 | 2018-02-13 00:25:36 | click FNOXAibRjkIaQPB |
| 1000235 | 2018-02-13 22:30:00 | click GczrYaxvkiPultZ |
| 1335656 | 2018-02-13 18:30:00 | click MxONdLckpAFUHRS, pay PEqMSHyktn |
| 1335656 | 2018-02-13 19:54:00 | click mKRPGOciFDyzTgM, pay PEqMSHyktn |
| 2656199 | 2018-02-13 08:30:00 | click CZwafHsbJOPNitL, pay pYvotuLDIT |
| 2656199 | 2018-02-13 09:14:00 | click nYHJqIpjevkKToy, pay pYvotuLDIT |
| 2656199 | 2018-02-13 21:05:00 | click gbAfPCwrGXvEjpI, pay PEqMSHyktn |
| 2656199 | 2018-02-13 21:08:00 | click dhpZyWMuGjBOTJP, pay PEqMSHyktn |
| 2656199 | 2018-02-13 22:29:00 | click bAsxnUdDhvfqaBr, pay gZhvdySOQb |
| 2656199 | 2018-02-13 22:30:00 | click XIhZdLaOocQRmrY, pay gZhvdySOQb |
| 4356142 | 2018-02-13 18:30:00 | click DYqShmGbIoWKier |
| 4356142 | 2018-02-13 19:54:00 | click DYqShmGbIoWKier |
| 8881237 | 2018-02-13 00:30:00 | click MpkvilgWSmhUuPn, pay pYvotuLDIT |
| 8881237 | 2018-02-13 06:14:00 | click OkTYNUHMqZzlDyL, pay pYvotuLDIT |
| 8881237 | 2018-02-13 10:30:00 | click OkTYNUHMqZzlDyL, pay KBuMzRpsko |
| 9890100 | 2018-02-13 16:01:00 | click vOTQfBFjcgXisYU, pay gZhvdySOQb |
| 9890100 | 2018-02-13 16:20:00 | click WxaLgOCcVEvhiFJ, pay MxONdLckwa |
+---------+------------+---------+
```

As shown in the preceding code, the task that could not be performed by calling native JOIN methods has been completed by using UDJ.

1.6.10.2.5. Pre-sorting

An iterator is used to search all records in the payment table and locate payment records that match the query. To perform this task, you must load all payment records with the same user_id to an ArrayList. This method can be applied when the number of payment records is small. Due to RAM size limits, you must find another method to load the data if a large number of payment records have been generated.

This topic describes how to address this issue using the SORT BY clause. When the size of the payment data is too large to be stored in the memory, it would be easier to address this issue if all data in the table has already been sorted by time. You then only need to compare the first element in these two lists. UDJ code in Java:

```
@Override
public void join(Record key, Iterator<Record> left, Iterator<Record> right, Yieldable<Record> output) {
  outputRecord.setString(0, key.getString(0));
  if (! right.hasNext()) {
    return;
  } else if (! left.hasNext()) {
    while (right.hasNext()) {
```

```
      Record logRecord = right.next();
      outputRecord.setBigint(1, logRecord.getDatetime(0).getTime());
      outputRecord.setString(2, logRecord.getString(1));
      output.yield(outputRecord);
    }
    return;
  }
  long prevDelta = Long.MAX_VALUE;
  Record logRecord = right.next();
  Record payRecord = left.next();
  Record lastPayRecord = payRecord.clone();
  while (true) {
    long delta = logRecord.getDatetime(0).getTime() - payRecord.getDatetime(0).getTime();
    if (left.hasNext() && delta > 0) {
      // The delta of time between two records is decreasing, we can still
      // explore the left group to try to gain a smaller delta.
      lastPayRecord = payRecord.clone();
      prevDelta = delta;
      payRecord = left.next();
    } else {
      // Hit to the point of minimal delta. Check with the last pay record,
      // output the merge result and prepare to process the next record of
      // right group.
      Record nearestPay = Math.abs(delta) < prevDelta ? payRecord : lastPayRecord;
      outputRecord.setBigint(1, logRecord.getDatetime(0).getTime());
      String mergedString = mergeLog(nearestPay.getString(1), logRecord.getString(1));
      outputRecord.setString(2, mergedString);
      output.yield(outputRecord);
      if (right.hasNext()) {
        logRecord = right.next();
        prevDelta = Math.abs(
          logRecord.getDatetime(0).getTime() - lastPayRecord.getDatetime(0).getTime()
        );
      } else {
        break;
      }
    }
  }
}
```

> **Notice** After you have modified the UDJ code, you must update the corresponding JAR package.

When the created UDJ function is used in MaxCompute SQL, you must modify the command as follows:

```
SELECT r.user_id, from_unixtime(time/1000) as time, content FROM (
 SELECT user_id, time as time, pay_info FROM payment
) p JOIN (
 SELECT user_id, time as time, content FROM user_client_log
) u
ON p.user_id = u.user_id
USING pay_user_log_merge_join(p.time, p.pay_info, u.time, u.content)
r
AS (user_id, time, content)
SORT BY p.time, u.time
;
```

In the native SQL language, you must make a few modifications, add a SORT BY clause to the end of the UDJ clause, and then sort the data in both tables by time.

The execution result is the same as the result before the code is modified.

This method uses the SORT BY clause to pre-sort the data. To achieve the same result, only a maximum of three records need to be cached.

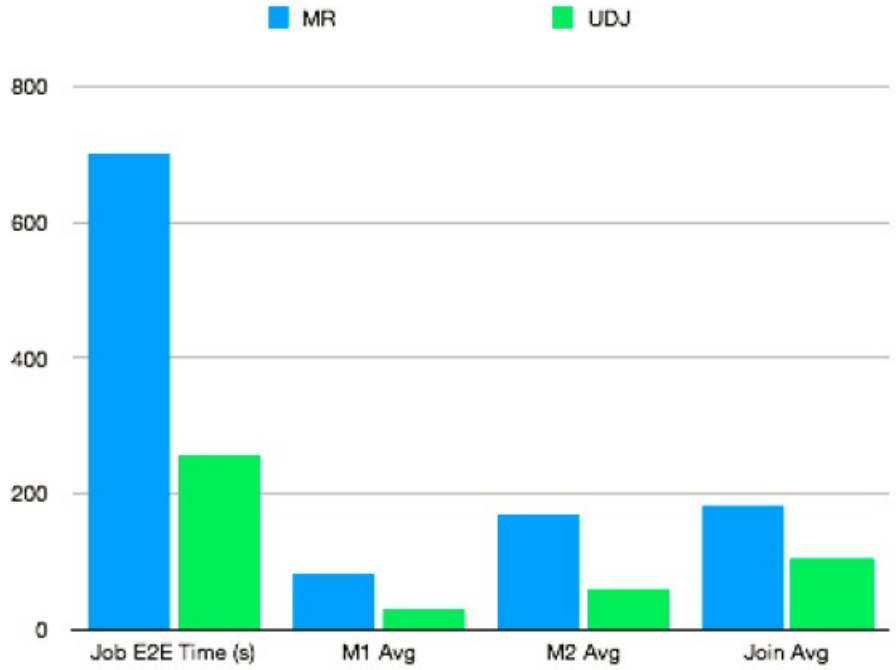
1.6.10.3. Performance advantages

Without UDJ, you must use MapReduce to handle complex cross-table computing tasks in a distributed system.

The following example uses an online MapReduce job to test the UDJ performance. This MapReduce job uses a complex algorithm to join two tables. This example uses UDJ to rewrite the SQL statements of the MapReduce job and checks the execution results.

Under the same programming concurrency, the comparison of performance is as follows.

Performance comparison



As shown in the figure, UDJ helps describe the complex logic of handling multiple tables, and greatly improves the query performance.

> **Note** The code is only executed inside UDJ. The entire logic of the code is executed by the high-performance MaxCompute native runtime.

UDJ optimizes the MaxCompute runtime engine and the data exchange between interfaces. The join logic of UDJ is more efficient than that of the reduce stage.

1.6.11. Parameterized view

MaxCompute supports parameterized views. You can call such views to obtain queried information.

In the traditional views of MaxCompute, complex SQL scripts are encapsulated at the underlying layer. Callers can call views like reading a standard table without the need to understand the underlying implementation mechanism. Traditional views are widely used because they can be used to implement encapsulation and code reuse.

However, you cannot specify parameters for traditional views. If a traditional view is called to read data from an underlying table, you cannot filter data in the underlying table or pass other parameters to the view. This reduces the code reuse rate.

The new SQL engine of MaxCompute V2.0 supports parameterized views and allows you to import any tables or other variables to customize views.

Create a parameterized view

Example

```
--view with parameters
-- param @a -a table parameter
-- param @b -a string parmeter
--returns a table with schema (key string,value string)
create view if not exists pv1(@a table (k string,v bigint), @b string)
as
select srcp.key,srcp.value from srcp join @a on srcp.key=a.k and srcp.p=@b;
```

- The created pv1 view has two parameters, the table and string parameters. The parameter values can be tables or be of a basic data type.
- The parameter values can also be subqueries. Example:

  ```
  select * from view_name( (select 1 fromsrc where a > 0), 1);
  ```

- When you define a view, you can set the type of a parameter value to ANY. Example:

  ```
  create view view_name (@a ANY, @b TABLE (x ANY)) as ...
  ```

- When you define a view, you can use an asterisk (*) to indicate a varying-length column. Example:

  ```
  create view view_name(@a bigint @b TABLE(x bigint, * ANY)) asselect * from @b where x = @a;
  ```

  Note In the table specified by the TABLE parameter, data in the first column is of the BIGINT type. You can execute `SELECT *` to obtain the varying-length part.

Call the created view

Execute the following statement to call the created pv1 view:

```
@a := select * from src where value >0;
--call view with table varable an scalar
@b := select * from pv1(@a,'20170101');
@another_day := '20170102';
--call view with table name and scalar variable
@c := select * from pv1(src2, @another_day);
@d := select * from @c union all select * from @b;
with
t as(select * from src3)
select * from @c
union all
select * from @d
union all
select * from pv1(t,@another_day);
```

Note

- You can use different parameters to call the pv1 view. The value of the table parameter can be a physical table, view, table variable, or table alias in common table expressions (CTEs).
- Common parameters can be variables or constants.

Additional instructions

A parameterized view can contain multiple SQL statements, similar to a script.

```
--view with parameters
-- param @a -a table parameter
-- param @b -a string parmeter
--returns a table with schema (key string,value string)
create view if not exists pv2(@a table (k string,v bigint), @b string) as
BEGIN
@srcp := select * from srcp where p=@b;
@pv2 := select srcp.key,srcp.value from @srcp join @a on srcp.key=a.k;
end;
```

Note

- Content between BEGIN and end is the script of this view.
- The `@pv2 := ...` statement is similar to the RETURN statement in other programming languages. This statement is used to assign a value to an implicit table variable that has the same name as the view.
- Only DML statements can be used in scripts. The INSERT and CREATE TABLE AS statements cannot be included in scripts.
- PRINT statements cannot be included in scripts.

The matching rules for actual and formal view parameters are the same as those specified in a normal programming language. If view parameters can be implicitly converted, these parameters can be matched. For example, the BIGINT value can match the parameters of the DOUBLE type. For table variables, if the schema of Table A can be inserted into Table B, Table A can be used to match the table parameter that has the same schema as Table B.

In some situations, you can declare the return type to make the code easier to read. Example:

```
create view if not exists pv3(@a table (k string,v bigint), @b string)
returns @ret table (x string,y string)
AS
begin
  @srcp := select * from srcp where p=@b;
  @ret := select srcp.key,srcp.value from @srcp join @a on srcp.key=a.k;
end;
```

> **Note** RETURNS @ret TABLE (x string, y string) defines the following information:
>
> - The return type is TABLE (x string, y string), which indicates the type returned to the caller. You can use this parameter to customize the table schema.
> - The response parameter is @ret. A value is assigned to the parameter in the view script.
> - You can regard a view that does not contain the BEGIN and END keywords or that does not return variables as a simplified view.

1.6.12. Geographic functions

1.6.12.1. Usage notes

Before you use geographic functions, understand the following points:

All functions are published in the geospatial project of the DataWorks marketplace. These functions are prefixed with ST_. You can click a function to view and use it without the need to apply for permissions. To use a function, add `geospatial.Project prefix` to the beginning of the function name and commit SQL statements that contain this function with the following flags:

```
set odps.sql.hive.compatible=true;
set odps.sql.udf.java.retain.legacy=false;
set odps.isolation.session.enable=true;
```

1.6.12.2. Constructors

1.6.12.2.1. ST_AsBinary

Function declaration:

```
ST_AsBinary(ST_Geometry)
```

Description: This function returns the well-known binary (WKB) representation of the input geometry.

Example:

```
SELECT ST_AsBinary(ST_Point(1, 2)) FROM onerow;
-- WKB representation of POINT (1 2)
```

1.6.12.2.2. ST_AsGeoJson

Function declaration:

```
ST_AsGeoJson(geometry)
```

Description: This function returns the GeoJSON representation of the input geometry.

Example:

```
SELECT ST_AsGeoJson(ST_Point(1.0, 2.0)) from onerow;
-- {"type":"Point", "coordinates":[1.0, 2.0]}
```

1.6.12.2.3. ST_AsJson

Function declaration:

```
ST_AsJSON(ST_Geometry)
```

Description: This function returns the JSON representation of the input geometry.

Example:

```
SELECT ST_AsJSON(ST_Point(1.0, 2.0)) from onerow;
-- {"x":1.0,"y":2.0}
SELECT ST_AsJSON(ST_SetSRID(ST_Point(1, 1), 4326)) from onerow;
-- {"x":1.0,"y":1.0,"spatialReference":{"wkid":4326}}
```

1.6.12.2.4. ST_AsShape

Function declaration:

```
ST_AsShape(ST_Geometry)
```

Description: This function returns the ESRI shape representation of the input geometry.

Example:

```
SELECT ST_AsShape(ST_Point(1, 2)) FROM onerow;
-- Esri shape representation of POINT (1 2)
```

1.6.12.2.5. ST_AsText

Function declaration:

```
ST_AsText(ST_Geometry)
```

Description: This function returns the well-known text (WKT) representation of the input geometry.

Example:

```
SELECT ST_AsText(ST_Point(1, 2)) FROM onerow;
-- POINT (1 2)
```

1.6.12.2.6. ST_GeomCollection

Function declaration:

```
ST_GeomCollection(wkt)
```

Description: This function constructs a multi-part geometry from the well-known text (WKT) representation based on the Open Geospatial Consortium (OGC).

> **Notice** The ST_GeomCollection function in MaxCompute supports only the multi-part geometry feature, not the collection feature.

Example:

```
SELECT ST_GeomCollection('multipoint ((1 0), (2 3))') FROM src LIMIT 1;
-- Construct a multipoint geometry.
ST_GeomCollection('POINT(1 1), LINESTRING(2 0,3 0)')
-- Not supported.
```

1.6.12.2.7. ST_GeomFromGeoJson

Function declaration:

```
ST_GeomFromGeoJson(json)
```

Description: This function constructs a geometry from the input GeoJSON representation.

Example:

```
SELECT ST_GeomFromGeoJson('{"type":"Point", "coordinates":[1.2, 2.4]}') FROM src LIMIT 1;
-- Construct a point.
SELECT ST_GeomFromGeoJson('{"type":"LineString", "coordinates":[[1,2], [3,4]]}') FROM src LIMIT 1;
-- Construct a linestring.
```

1.6.12.2.8. ST_GeomFromJSON

Function declaration:

```
ST_GeomFromJSON(json)
```

Description: This function constructs a geometry from the input ESRI JSON representation.

Example:

```
SELECT ST_GeomFromJSON('{"x":0.0,"y":0.0}') FROM src LIMIT 1;
-- Construct a point.
```

1.6.12.2.9. ST_GeomFromShape

Function declaration:

```
ST_GeomFromShape(shape)
```

Description: This function constructs a geometry from the input ESRI shape representation.

Example:

```
SELECT ST_GeomFromShape(ST_AsShape(ST_Point(1, 2)));
-- Construct a point.
```

1.6.12.2.10. ST_GeomFromText

Function declaration:

```
ST_GeomFromText(wkt)
```

Description: This function constructs a geometry from the input well-known text (WKT) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_GeomFromText('linestring (1 0, 2 3)') FROM src LIMIT 1;
-- Construct a linestring.
SELECT ST_GeomFromText('multipoint ((1 0), (2 3))') FROM src LIMIT 1;
-- Construct a multipoint geometry.
```

1.6.12.2.11. ST_GeomFromWKB

Function declaration:

```
ST_GeomFromWKB(wkb)
```

Description: This function constructs a geometry from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_GeomFromWKB(ST_AsBinary(ST_GeomFromText('linestring (1 0, 2 3)'))) FROM src LIMIT 1;
-- Construct a linestring.
SELECT ST_GeomFromWKB(ST_AsBinary(ST_GeomFromText('multipoint ((1 0), (2 3))'))) FROM src LIMIT 1
;
-- Construct a multipoint geometry.
```

1.6.12.2.12. ST_GeometryType

Function declaration:

```
ST_GeometryType(geometry)
```

Description: This function returns the type name of the input geometry.

Example:

```
SELECT ST_GeometryType(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- ST_Point
SELECT ST_GeometryType(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- ST_LineString
SELECT ST_GeometryType(ST_Polygon(2,0, 2,3, 3,0)) FROM src LIMIT 1;
-- ST_Polygon
```

1.6.12.2.13. ST_LineString

Function declaration:

```
ST_LineString(x, y, [x, y]*)
ST_LineString('linestring( ... )')
ST_LineString(array(x+), array(y+))
ST_LineString(array(ST_Point(x,y)+)))
```

Description: This function constructs a two-dimensional line.

Example:

```
SELECT ST_LineString(1, 1, 2, 2, 3, 3) from src LIMIT 1;
SELECT ST_LineString('linestring(1 1, 2 2, 3 3)') from src LIMIT 1;
SELECT ST_LineString(array(1,2,3), array (1,2,3)) from src LIMIT 1;
SELECT ST_LineString(array(ST_Point(1, 1), ST_Point(2,2), ST_Point(3,3))) from src LIMIT 1;
```

1.6.12.2.14. ST_LineFromWKB

Function declaration:

```
ST_LineFromWKB(wkb)
```

Description: This function constructs a two-dimensional line from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_LineFromWKB(ST_AsBinary(ST_GeomFromText('linestring (1 0, 2 3)'))) FROM src LIMIT 1;
-- Construct a two-dimensional line.
```

1.6.12.2.15. ST_MultiLineString

Function declaration:

```
ST_MultiLineString(array(x1, y1, x2, y2, ... ), array(x1, y1, x2, y2, ... ), ... )
ST_MultiLineString('multilinestring( ... )')
```

Description: This function constructs a two-dimensional multilinestring.

Example:

```
SELECT ST_MultiLineString(array(1, 1, 2, 2), array(10, 10, 20, 20)) from src LIMIT 1;
SELECT ST_MultiLineString('multilinestring ((1 1, 2 2), (10 10, 20 20))', 0) from src LIMIT 1;
```

1.6.12.2.16. ST_MLineFromWKB

Function declaration:

```
ST_MLineFromWKB(wkb)
```

Description: This function constructs a two-dimensional multilinestring from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_MLineFromWKB(ST_AsBinary(ST_GeomFromText('multilinestring ((1 0, 2 3), (5 7, 7 5))'))) FRO
M src LIMIT 1;
-- Construct a two-dimensional multilinestring.
```

1.6.12.2.17. ST_MultiPoint

Function declaration:

```
ST_MultiPoint(x1, y1, x2, y2, x3, y3)
ST_MultiPoint('multipoint( ... )')
```

Description: This function constructs a two-dimensional multipoint geometry.

Example:

```
SELECT ST_MultiPoint(1, 1, 2, 2, 3, 3) from src LIMIT 1;
-- Construct a three-point geometry.
SELECT ST_MultiPoint('MULTIPOINT ((10 40), (40 30))') from src LIMIT 1;
-- Construct a two-point geometry.
```

1.6.12.2.18. ST_MPointFromWKB

Function declaration:

```
ST_MPointFromWKB(wkb)
```

Description: This function constructs a two-dimensional multipoint geometry from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_MPointFromWKB(ST_AsBinary(ST_GeomFromText('multipoint ((1 0), (2 3))'))) FROM src LIMIT
1;
-- Construct a two-dimensional multipoint geometry.
```

1.6.12.2.19. ST_MultiPolygon

Function declaration:

```
ST_MultiPolygon(array(x1, y1, x2, y2, ... ), array(x1, y1, x2, y2, ... ), ... )
ST_MultiPolygon('multipolygon ( ... )')
```

Description: This function constructs a two-dimensional multipolygon.

Example:

```
SELECT ST_MultiPolygon(array(1, 1, 1, 2, 2, 2, 2, 1), array(3, 3, 3, 4, 4, 4, 4, 3)) from src LIMIT 1;
SELECT ST_MultiPolygon('multipolygon (((0 0, 0 1, 1 0, 0 0)), ((2 2, 2 3, 3 2, 2 2)))') from src LIMIT 1;
```

1.6.12.2.20. ST_MPolyFromWKB

Function declaration:

```
ST_MPolyFromWKB(wkb)
```

Description: This function constructs a two-dimensional multipolygon from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_MPolyFromWKB(ST_AsBinary(ST_GeomFromText('multipolygon (((0 0, 1 0, 0 1, 0 0)), ((2 2, 1 2,
2 1, 2 2)))'))) FROM src LIMIT 1;
-- Construct a two-dimensional multipolygon.
```

1.6.12.2.21. ST_Point

Function declaration:

```
ST_Point(x, y)
ST_Point('point (x y)')
```

Description: This function constructs a two-dimensional point.

Example:

```
SELECT ST_Point(longitude, latitude) from src LIMIT 1;
SELECT ST_Point('point (0 0)') from src LIMIT 1;
```

1.6.12.2.22. ST_PointFromWKB

Function declaration:

```
ST_PointFromWKB(wkb)
```

Description: This function constructs a two-dimensional point from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_PointFromWKB(ST_AsBinary(ST_GeomFromText('point (1 0)'))) FROM src LIMIT 1;
-- Construct a two-dimensional point.
```

1.6.12.2.23. ST_PointZ

Function declaration:

```
ST_PointZ(x, y, z)
```

Description: This function constructs a three-dimensional point.

Example:

```
SELECT ST_PointZ(longitude, latitude, elevation) from src LIMIT 1;
```

1.6.12.2.24. ST_Polygon

Function declaration:

```
ST_Polygon(x, y, [x, y]*)
ST_Polygon('polygon( ... )')
```

Description: This function constructs a two-dimensional polygon.

Example:

```
SELECT ST_Polygon(1, 1, 1, 4, 4, 4, 4, 1) from src LIMIT 1;
-- Construct a square.
SELECT ST_Polygon('polygon ((1 1, 4 1, 1 4))') from src LIMIT 1;
-- Construct a triangle.
```

1.6.12.2.25. ST_PolyFromWKB

Function declaration:

```
ST_PolyFromWKB(wkb)
```

Description: This function constructs a two-dimensional polygon from the input well-known binary (WKB) representation based on the Open Geospatial Consortium (OGC).

Example:

```
SELECT ST_PolyFromWKB(ST_AsBinary(ST_GeomFromText('polygon ((0 0, 10 0, 0 10, 0 0))'))) FROM src LI
MIT 1;
-- Construct a two-dimensional polygon.
```

1.6.12.2.26. ST_SetSRID

Function declaration:

```
ST_SetSRID(<ST_Geometry>, SRID)
```

Description: This function sets the spatial reference system identifier (SRID) of the input geometry.

Example:

```
SELECT ST_SetSRID(ST_Point(1.5, 2.5), 4326)) FROM src LIMIT 1;
-- Construct a point and set its SRID to 4326.
```

1.6.12.3. Accessors

1.6.12.3.1. ST_Area

Function declaration:

```
ST_Area(ST_Polygon)
```

Description: This function returns the areas of one or more polygons.

Example:

```
SELECT ST_Area(ST_Polygon(1,1, 1,4, 4,4, 4,1)) FROM src LIMIT 1;
-- 9.0
```

1.6.12.3.2. ST_Centroid

Function declaration:

```
ST_Centroid(polygon)
```

Description: This function returns the center point of the minimum bounding rectangle of the input polygon.

Example:

```
SELECT ST_Centroid(ST_GeomFromText('polygon ((0 0, 3 6, 6 0, 0 0))')) FROM src LIMIT 1;
-- POINT(3 3)
SELECT ST_Centroid(ST_GeomFromText('polygon ((0 0, 0 8, 8 0, 0 0))')) FROM src LIMIT 1;
-- POINT(4 4)
```

1.6.12.3.3. ST_CoordDim

Function declaration:

```
ST_CoordDim(geometry)
```

Description: This function returns the coordinate dimension of the input geometry.

Example:

```
SELECT ST_CoordDim(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 2
SELECT ST_CoordDim(ST_PointZ(1.5,2.5, 3) FROM src LIMIT 1;
-- 3
SELECT ST_CoordDim(ST_Point(1.5, 2.5, 3., 4.)) FROM src LIMIT 1;
-- 4
```

1.6.12.3.4. ST_Dimension

Function declaration:

```
ST_Dimension(geometry)
```

Description: This function returns the spatial dimension of the input geometry.

Example:

```
SELECT ST_Dimension(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 0
SELECT ST_Dimension(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 1
SELECT ST_Dimension(ST_Polygon(2,0, 2,3, 3,0)) FROM src LIMIT 1;
-- 2
```

1.6.12.3.5. ST_Distance

Function declaration:

```
ST_Distance(ST_Geometry1, ST_Geometry2)
```

Description: This function returns the distance between a point in geometry1 and a point in geometry2.

Example:

```
SELECT ST_Distance(ST_Point(0.0,0.0), ST_Point(3.0,4.0)) FROM src LIMIT 1;
-- 5.0
```

1.6.12.3.6. ST_GeodesicLengthWGS84

Function declaration:

```
ST_GeodesicLengthWGS84(line)
```

Description: This function returns the distance in meters on a spheroid based on World Geodetic System 1984 (WGS84). The geometry must be in WGS84. Otherwise, this function returns NULL.

Example:

```
SELECT ST_GeodesicLengthWGS84(ST_SetSRID(ST_Linestring(0.0,0.0, 0.3,0.4), 4326)) FROM src LIMIT 1;
-- 55km
SELECT ST_GeodesicLengthWGS84(ST_GeomFromText('MultiLineString((0.0 80.0, 0.3 80.4))', 4326)) FROM src LIMIT 1;
-- 45km
```

1.6.12.3.7. ST_GeometryN

Function declaration:

```
ST_GeometryN(ST_GeometryCollection, n)
```

Description: This function returns the nth geometry in the input geometry collection. n starts from 1.

Example:

```
SELECT ST_GeometryN(ST_GeomFromText('multipoint ((10 40), (40 30), (20 20), (30 10))'), 3) FROM src LIMIT 1;
-- ST_Point(20 20)
SELECT ST_GeometryN(ST_GeomFromText('multilinestring ((2 4, 10 10), (20 20, 7 8))'), 2) FROM src LIMIT 1;
-- ST_Linestring(20 20, 7 8)
```

1.6.12.3.8. ST_Is3D

Function declaration:

```
ST_Is3D(geometry)
```

Description: If the input geometry has Z coordinates, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Is3D(ST_Polygon(1,1, 1,4, 4,4, 4,1)) FROM src LIMIT 1;
-- false
SELECT ST_Is3D(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- false
SELECT ST_Is3D(ST_Point(3., 4.)) FROM src LIMIT 1;
-- false
SELECT ST_Is3D(ST_PointZ(3., 4., 2)) FROM src LIMIT 1;
-- true
```

1.6.12.3.9. ST_IsClosed

Function declaration:

```
ST_IsClosed(ST_[Multi]LineString)
```

Description: If the input linestring or linestrings are closed, this function returns true.

Example:

```
SELECT ST_IsClosed(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- true
SELECT ST_IsClosed(ST_LineString(0.,0., 3.,4.)) FROM src LIMIT 1;
-- false
```

1.6.12.3.10. ST_IsEmpty

Function declaration:

```
ST_IsEmpty(geometry)
```

Description: If the input geometry is empty, this function returns true.

Example:

```
SELECT ST_IsEmpty(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- false
SELECT ST_IsEmpty(ST_GeomFromText('point empty')) FROM src LIMIT 1;
-- true
```

1.6.12.3.11. ST_IsMeasured

Function declaration:

```
ST_IsMeasured(geometry)
```

Description: If the input geometry has M coordinates (measures), this function returns true.

Example:

```
SELECT ST_IsMeasured(ST_Polygon(1,1, 1,4, 4,4, 4,1)) FROM src LIMIT 1;
-- false
SELECT ST_IsMeasured(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- false
SELECT ST_IsMeasured(ST_Point(3., 4.)) FROM src LIMIT 1;
-- false
SELECT ST_IsMeasured(ST_PointM(3., 4., 2)) FROM src LIMIT 1;
-- true
```

1.6.12.3.12. ST_IsSimple

Function declaration:

```
ST_IsSimple(geometry)
```

Description: If the input geometry is simple, this function returns true.

Example:

```
SELECT ST_IsSimple(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- true
SELECT ST_IsSimple(ST_LineString(0.,0., 1.,1., 0.,1., 1.,0.)) FROM src LIMIT 1;
-- false
```

1.6.12.3.13. ST_IsRing

Function declaration:

```
ST_IsRing(ST_LineString)
```

Description: If the input linestring is closed or simple, this function returns true.

Example:

```
SELECT ST_IsRing(ST_LineString(0.,0., 3.,4., 0.,4., 0.,0.)) FROM src LIMIT 1;
-- true
SELECT ST_IsRing(ST_LineString(0.,0., 1.,1., 1.,2., 2.,1., 1.,1., 0.,0.)) FROM src LIMIT 1;
-- false
SELECT ST_IsRing(ST_LineString(0.,0., 3.,4.)) FROM src LIMIT 1;
-- false
```

1.6.12.3.14. ST_Length

Function declaration:

```
ST_Length(line)
```

Description: This function returns the length of the input line segment.

Example:

```
SELECT ST_Length(ST_Line(0.0,0.0, 3.0,4.0)) FROM src LIMIT 1;
-- 5.0
```

1.6.12.3.15. ST_M

Function declaration:

```
ST_M(geometry)
```

Description: This function returns the M coordinate of the input geometry.

Example:

```
SELECT ST_M(ST_PointM(3., 4., 2)) FROM src LIMIT 1;
-- 2
```

1.6.12.3.16. ST_MaxM

Function declaration:

```
ST_MaxM(geometry)
```

Description: This function returns the maximum M coordinate of the input geometry.

Example:

```
SELECT ST_MaxM(ST_PointM(1.5, 2.5, 2)) FROM src LIMIT 1;
-- 2
SELECT ST_MaxM(ST_LineString('linestring m (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
-- 1
```

1.6.12.3.17. ST_MinM

Function declaration:

```
ST_MinM(geometry)
```

Description: This function returns the minimum M coordinate of the input geometry.

Example:

```
SELECT ST_MinM(ST_PointM(1.5, 2.5, 2)) FROM src LIMIT 1;
-- 2
SELECT ST_MinM(ST_LineString('linestring m (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
-- 1
```

1.6.12.3.18. ST_X

Function declaration:

```
ST_X(point)
```

Description: This function returns the X coordinate of the input point.

Example:

```
SELECT ST_X(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
--  1.5
```

1.6.12.3.19. ST_Y

Function declaration:

```
ST_Y(point)
```

Description: This function returns the Y coordinate of the input point.

Example:

```
SELECT ST_Y(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
--  2.5
```

1.6.12.3.20. ST_Z

Function declaration:

```
ST_Z(point)
```

Description: This function returns the Z coordinate of the input point.

Example:

```
SELECT ST_Z(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
--  1.5
```

1.6.12.3.21. ST_MaxX

Function declaration:

```
ST_MaxX(geometry)
```

Description: This function returns the maximum X coordinate of the input geometry.

Example:

```
SELECT ST_MaxX(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 1.5
SELECT ST_MaxX(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 3.0
```

1.6.12.3.22. ST_MaxY

Function declaration:

```
ST_MaxY(geometry)
```

Description: This function returns the maximum Y coordinate of the input geometry.

Example:

```
SELECT ST_MaxY(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 2.5
SELECT ST_MaxY(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 2.5
```

1.6.12.3.23. ST_MaxZ

Function declaration:

```
ST_MaxZ(geometry)
```

Description: This function returns the maximum Z coordinate of the input geometry.

Example:

```
SELECT ST_MaxZ(ST_PointZ(1.5, 2.5, 2)) FROM src LIMIT 1;
-- 2
SELECT ST_MaxZ(ST_LineString('linestring z (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
-- 1
```

1.6.12.3.24. ST_MinX

Function declaration:

```
ST_MinX(geometry)
```

Description: This function returns the minimum X coordinate of the input geometry.

Example:

```
SELECT ST_MinX(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 1.5
SELECT ST_MinX(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 3.0
```

1.6.12.3.25. ST_MinY

Function declaration:

```
ST_MinY(geometry)
```

Description: This function returns the minimum Y coordinate of the input geometry.

Example:

```
SELECT ST_MinY(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 2.5
SELECT ST_MinY(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 2.2
```

1.6.12.3.26. ST_MinZ

Function declaration:

```
ST_MinZ(geometry)
```

Description: This function returns the minimum Z coordinate of the input geometry.

Example:

```
SELECT ST_MinZ(ST_PointZ(1.5, 2.5, 2)) FROM src LIMIT 1;
-- 2
SELECT ST_MinZ(ST_LineString('linestring z (1.5 2.5 2, 3.0 2.2 1)')) FROM src LIMIT 1;
-- 1
```

1.6.12.3.27. ST_NumGeometries

Function declaration:

```
ST_NumGeometries(ST_GeometryCollection)
```

Description: This function returns the number of geometries in the input geometry collection.

Example:

```
SELECT ST_NumGeometries(ST_GeomFromText('multipoint ((10 40), (40 30), (20 20), (30 10))')) FROM src
LIMIT 1;
-- 4
SELECT ST_NumGeometries(ST_GeomFromText('multilinestring ((2 4, 10 10), (20 20, 7 8))')) FROM src LIM
IT 1;
-- 2
```

1.6.12.3.28. ST_NumInteriorRing

Function declaration:

```
ST_NumInteriorRing(ST_Polygon)
```

Description: This function returns the number of interior rings of the input polygon.

Example:

```
SELECT ST_NumInteriorRing(ST_Polygon(1,1, 1,4, 4,1)) FROM src LIMIT 1;
-- 0
SELECT ST_NumInteriorRing(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1 1))')) FROM src LIMIT
1;
-- 1
```

1.6.12.3.29. ST_NumPoints

Function declaration:

```
ST_NumPoints(geometry)
```

Description: This function returns the number of points in the input geometry.

Example:

```
SELECT ST_NumPoints(ST_Point(1.5, 2.5)) FROM src LIMIT 1;
-- 1
SELECT ST_NumPoints(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- 2
SELECT ST_NumPoints(ST_GeomFromText('polygon ((0 0, 10 0, 0 10, 0 0))')) FROM src LIMIT 1;
-- 4
```

1.6.12.3.30. ST_PointN

Function declaration:

```
ST_PointN(ST_Geometry, n)
```

Description: This function returns the nth point of one or more linestrings.

Example:

```
SELECT ST_PointN(ST_LineString(1.5,2.5, 3.0,2.2), 2) FROM src LIMIT 1;
-- POINT(3.0 2.2)
```

1.6.12.3.31. ST_StartPoint

Function declaration:

```
ST_StartPoint(geometry)
```

Description: This function returns the first point of the input linestring.

Example:

```
SELECT ST_StartPoint(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- POINT(1.5 2.5)
```

1.6.12.3.32. ST_EndPoint

Function declaration:

```
ST_EndPoint(geometry)
```

Description: This function returns the last point of the input linestring.

Example:

```
SELECT ST_EndPoint(ST_LineString(1.5,2.5, 3.0,2.2)) FROM src LIMIT 1;
-- POINT(3.0 2.0)
```

1.6.12.3.33. ST_SRID

Function declaration:

```
ST_SRID(ST_Geometry)
```

Description: This function returns the spatial reference system identifier (SRID) of the input geometry.

Example:

```
SELECT ST_SRID(ST_Point(1.5, 2.5)) FROM src LIMIT 1
-- Return SRID 0.
```

1.6.12.4. Operations

1.6.12.4.1. ST_Aggr_ConvexHull

Function declaration:

```
ST_Aggr_ConvexHull(ST_Geometry)
```

Description: This function returns a convex hull for input geometries by using aggregation transformation.

Example:

```
SELECT ST_Aggr_ConvexHull(geometry) FROM source;
-- Return the convex hull of input geometries from the data source by using aggregation transformati
on.
```

1.6.12.4.2. ST_Aggr_Intersection

Function declaration:

```
ST_Aggr_Intersection(ST_Geometry)
```

Description: This function returns the intersection of input geometries by using aggregation transformation.

Example:

```
SELECT ST_Aggr_Intersection(geometry) FROM source;
-- Return the intersection of input geometries from the data source by using aggregation transformati
on.
```

1.6.12.4.3. ST_Aggr_Union

Function declaration:

```
ST_Aggr_Union(ST_Geometry)
```

Description: This function returns a union of input geometries by using aggregation transformation.

Example:

```
SELECT ST_Aggr_Union(geometry) FROM source;
-- Return the union of input geometries from the data source by using aggregation transformation.
```

1.6.12.4.4. ST_Bin

Function declaration:

```
ST_Bin(placeholder)
```

Description: This function returns the bin ID of the input point.

1.6.12.4.5. ST_BinEnvelope

Function declaration:

```
ST_BinEnvelope(binsize, point)
```

Description: This function returns the binary envelope for the input point.

Function declaration:

```
ST_BinEnvelope(binsize, binid)
```

Description: This function returns the binary envelope for the input bin ID.

1.6.12.4.6. ST_Boundary

Function declaration:

```
ST_Boundary(ST_Geometry)
```

Description: This function returns the boundary of the input geometry.

Example:

```
SELECT ST_Boundary(ST_LineString(0,1, 1,0))) FROM src LIMIT 1;
-- MULTIPOINT((1 0),(0 1))
SELECT ST_Boundary(ST_Polygon(1,1, 4,1, 1,4)) FROM src LIMIT 1;
-- LINESTRING(1 1, 4 1, 1 4, 1 1)
```

1.6.12.4.7. ST_Buffer

Function declaration:

```
ST_Buffer(geometry, distance)
```

Description: This function returns a geometry that indicates all points whose distance from this geometry to the input geometry is less than or equal to the value of the distance parameter.

1.6.12.4.8. ST_ConvexHull

Function declaration:

```
ST_ConvexHull(ST_Geometry, ST_Geometry, ...)
```

Description: This function returns the convex hull of the input geometry.

Example:

```
SELECT ST_AsText(ST_ConvexHull(ST_Point(0, 0), ST_Point(0, 1), ST_Point(1, 1))) FROM onerow;
-- MULTIPOLYGON (((0 0, 1 1, 0 1, 0 0)))
```

1.6.12.4.9. ST_Difference

Function declaration:

```
ST_Difference(ST_Geometry1, ST_Geometry2)
```

Description: This function returns a geometry that indicates the difference between the input geometries.

Example:

```
SELECT ST_AsText(ST_Difference(ST_MultiPoint(1, 1, 1.5, 1.5, 2, 2), ST_Point(1.5, 1.5))) FROM onerow;
-- MULTIPOINT (1 1, 2 2)
SELECT ST_AsText(ST_Difference(ST_Polygon(0, 0, 0, 10, 10, 10, 10, 0), ST_Polygon(0, 0, 0, 5, 5, 5, 5, 0))) f
rom onerow;
-- MULTIPOLYGON (((10 0, 10 10, 0 10, 0 5, 5 5, 5 0, 10 0)))
```

1.6.12.4.10. ST_Envelope

Function declaration:

```
ST_Envelope(ST_Geometry)
```

Description: This function returns the envelope of the input geometry. If the specified geometry is a point, a horizontal line, or a vertical line, this function returns the common difference or an empty envelope.

Example:

```
SELECT ST_Envelope(ST_LineString(0,0, 2,2)) from src LIMIT 1;
-- POLYGON ((0 0, 2 0, 2 2, 0 2, 0 0))
SELECT ST_Envelope(ST_Polygon(2,0, 2,3, 3,0)) from src LIMIT 1;
-- POLYGON ((2 0, 3 0, 3 3, 2 3, 2 0))
```

1.6.12.4.11. ST_ExteriorRing

Function declaration:

```
ST_ExteriorRing(polygon)
```

Description: This function returns the exterior ring of a polygon as a linestring.

Example:

```
SELECT ST_ExteriorRing(ST_Polygon(1,1, 1,4, 4,1)) FROM src LIMIT 1;
-- LINESTRING(1 1, 4 1, 1 4, 1 1)
SELECT ST_ExteriorRing(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1 1))')) FROM src LIMIT 1;
-- LINESTRING (8 0, 0 8, 0 0, 8 0)
```

1.6.12.4.12. ST_InteriorRingN

Function declaration:

```
ST_InteriorRingN(ST_Polygon, n)
```

Description: This function returns the nth interior ring of a polygon as a linestring.

Example:

```
SELECT ST_InteriorRingN(ST_Polygon('polygon ((0 0, 8 0, 0 8, 0 0), (1 1, 1 5, 5 1, 1 1))'), 1) FROM src LIMIT
1;
-- LINESTRING (1 1, 5 1, 1 5, 1 1)
```

1.6.12.4.13. ST_Intersection

Function declaration:

```
ST_Intersection(ST_Geometry1, ST_Geometry2)
```

Description: This function returns a geometry that indicates the intersection of the input geometries. If the input geometries intersect in a lower dimension, ST_Intersection may drop lower-dimension intersections or return a closed linestring.

Example:

```
SELECT ST_AsText(ST_Intersection(ST_Point(1,1), ST_Point(1,1))) FROM onerow;
-- POINT (1 1)
SELECT ST_AsText(ST_Intersection(ST_GeomFromText('linestring(0 2, 0 0, 2 0)'), ST_GeomFromText('lin
estring(0 3, 0 1, 1 0, 3 0)'))) FROM onerow;
-- MULTILINESTRING ((1 0, 2 0), (0 2, 0 1))
SELECT ST_AsText(ST_Intersection(ST_LineString(0,2, 2,3), ST_Polygon(1,1, 4,1, 4,4, 1,4))) FROM onerow
;
-- MULTILINESTRING ((1 2.5, 2 3))
SELECT ST_AsText(ST_Intersection(ST_Polygon(2,0, 2,3, 3,0), ST_Polygon(1,1, 4,1, 4,4, 1,4))) FROM onero
w;
-- MULTIPOLYGON (((2.67 1, 2 3, 2 1, 2.67 1)))
SELECT ST_AsText(ST_Intersection(ST_Polygon(2,0, 3,1, 2,1), ST_Polygon(1,1, 4,1, 4,4, 1,4))) FROM onero
w;
-- MULTIPOLYGON EMPTY or LINESTRING (2 1, 3 1, 2 1)
```

1.6.12.4.14. ST_SymmetricDiff

Function declaration:

```
ST_SymmetricDiff(ST_Geometry1, ST_Geometry2)
```

Description: This function returns a geometry that consists of the symmetric differences of the input geometries.

Example:

```
SELECT ST_AsText(ST_SymmetricDiff(ST_LineString('linestring(0 2, 2 2)'), ST_LineString('linestring(1 2, 3 2)'))) FROM onerow;
-- MULTILINESTRING((0 2, 1 2), (2 2, 3 2))
SELECT ST_AsText(ST_SymmetricDiff(ST_SymmetricDiff(ST_Polygon('polygon((0 0, 2 0, 2 2, 0 2, 0 0))'), ST_Polygon('polygon((1 1, 3 1, 3 3, 1 3, 1 1))'))) from onerow;
-- MULTIPOLYGON (((0 0, 2 0, 2 1, 1 1, 1 2, 0 2, 0 0)), ((3 1, 3 3, 1 3, 1 2, 2 2, 2 1, 3 1)))
```

1.6.12.4.15. ST_Union

Function declaration:

```
ST_Union(ST_Geometry, ST_Geometry, ...)
```

Description: This function returns a geometry that is the union of the input geometries.

Example:

```
SELECT ST_AsText(ST_Union(ST_Polygon(1, 1, 1, 4, 4, 4, 4, 1), ST_Polygon(4, 1, 4, 4, 4, 8, 8, 1))) FROM one row;
-- MULTIPOLYGON (((4 1, 8 1, 4 8, 4 4, 1 4, 1 1, 4 1)))
```

1.6.12.5. Relationship tests

1.6.12.5.1. ST_Contains

Function declaration:

```
BOOLEAN ST_Contains(geometry1, geometry2)
```

Description: If geometry1 contains geometry2, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Contains(st_polygon(1,1, 1,4, 4,4, 4,1), st_point(2, 3) from src LIMIT 1;
-- true is returned.
SELECT ST_Contains(st_polygon(1,1, 1,4, 4,4, 4,1), st_point(8, 8) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.2. ST_Crosses

Function declaration:

```
BOOLEAN ST_Crosses(geometry1, geometry2)
```

Description: If geometry1 crosses geometry2, this function returns true. Otherwise, this function returns false.

> **Note** Crossing indicates that some points in the two geometries are the same.

Example:

```
SELECT ST_Crosses(st_linestring(0,0, 1,1), st_linestring(1,0, 0,1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Crosses(st_linestring(2,0, 2,3), st_polygon(1,1, 1,4, 4,4, 4,1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Crosses(st_linestring(0,2, 0,1), ST_linestring(2,0, 1,0)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.3. ST_Disjoint

Function declaration:

```
BOOLEAN ST_Disjoint(geometry1, geometry2)
```

Description: If geometry1 and geometry2 do not intersect, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Disjoint(ST_LineString(0,0, 0,1), ST_LineString(1,1, 1,0)) from src LIMIT 1;
-- true is returned.
SELECT ST_Disjoint(ST_LineString(0,0, 1,1), ST_LineString(1,0, 0,1)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.4. ST_EnvIntersects

Function declaration:

```
BOOLEAN ST_EnvIntersects(ST_Geometry1, ST_Geometry2)
```

Description: If the envelopes of geometry1 and geometry2 intersect, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_EnvIntersects(ST_LineString(0,0, 1,1), ST_LineString(1,3, 2,2)) from src LIMIT 1;
-- false is returned.
SELECT ST_EnvIntersects(ST_LineString(0,0, 2,2), ST_LineString(1,0, 3,2)) from src LIMIT 1;
-- true is returned.
```

1.6.12.5.5. ST_Equals

Function declaration:

```
BOOLEAN ST_Equals(geometry1, geometry2)
```

Description: If geometry1 equals geometry2, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Equals(st_linestring(0,0, 1,1), st_linestring(1,1, 0,0)) from src LIMIT 1;
-- true is returned.
SELECT ST_Equals(st_linestring(0,0, 1,1), st_linestring(1,0, 0,1)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.6. ST_Intersects

Function declaration:

```
BOOLEAN ST_Intersects(geometry1, geometry2)
```

Description: If geometry1 and geometry2 intersect, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Intersects(st_linestring(0,0, 1,1), st_linestring(1,1, 0,0)) from src LIMIT 1;
-- true is returned.
SELECT ST_Intersects(st_linestring(0,0, 1,1), st_linestring(1,0, 0,1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Intersects(ST_LineString(2,0, 2,3), ST_Polygon(1,1, 4,1, 4,4, 1,4)) from src LIMIT 1;
-- true is returned.
SELECT ST_Intersects(ST_LineString(8,7, 7,8), ST_Polygon(1,1, 4,1, 4,4, 1,4)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.7. ST_Overlaps

Function declaration:

```
BOOLEAN ST_Overlaps(geometry1, geometry2)
```

Description: If geometry1 and geometry2 overlap, this function returns true. Otherwise, this function returns false. Overlapping excludes the tangency of the geometries.

Example:

```
SELECT ST_Overlaps(st_polygon(2,0, 2,3, 3,0), st_polygon(1,1, 1,4, 4,4, 4,1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Overlaps(st_polygon(2,0, 2,1, 3,1), ST_Polygon(1,1, 1,4, 4,4, 4,1)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.8. ST_Relate

Function declaration:

```
BOOLEAN ST_Relate(geometry1, geometry2)
```

Description: If geometry1 has the specified Dimensionally Extended nine-Intersection Model (DE-9IM) relationship with geometry2, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Relate(st_polygon(2,0, 2,1, 3,1), ST_Polygon(1,1, 1,4, 4,4, 4,1), '****T****') from src LIMIT 1;
-- true is returned.
SELECT ST_Relate(st_polygon(2,0, 2,1, 3,1), ST_Polygon(1,1, 1,4, 4,4, 4,1), 'T********') from src LIMIT 1;
-- false is returned.
SELECT ST_Relate(st_linestring(0,0, 3,3), ST_linestring(1,1, 4,4), 'T********') from src LIMIT 1;
-- true is returned.
SELECT ST_Relate(st_linestring(0,0, 3,3), ST_linestring(1,1, 4,4), '****T****') from src LIMIT 1;
-- false is returned.
```

1.6.12.5.9. ST_Touches

Function declaration:

```
BOOLEAN ST_Touches(geometry1, geometry2)
```

Description: If geometry1 and geometry2 spatially touch and have no similar interior points, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Touches(st_point(1, 2), st_polygon(1, 1, 1, 4, 4, 4, 4, 1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Touches(st_point(8, 8), st_polygon(1, 1, 1, 4, 4, 4, 4, 1)) from src LIMIT 1;
-- false is returned.
```

1.6.12.5.10. ST_Within

Function declaration:

```
BOOLEAN ST_Within(geometry1, geometry2)
```

Description: If geometry1 is within geometry2, this function returns true. Otherwise, this function returns false.

Example:

```
SELECT ST_Within(st_point(2, 3), st_polygon(1,1, 1,4, 4,4, 4,1)) from src LIMIT 1;
-- true is returned.
SELECT ST_Within(st_point(8, 8), st_polygon(1,1, 1,4, 4,4, 4,1)) from src LIMIT 1;
-- false is returned.
```

1.6.12.6. Geohash index functions

1.6.12.6.1. ST_GeoHash

Function declaration:

```
string ST_GeoHash(st_geometry geometry, integer precision=full_precision)
string ST_GeoHash(double longitude, double latitude, integer precision=full_precision)
```

Description: This function returns the unique Geohash string of the specified point. This function uses a function with the ST_ prefix or the specified longitudes and latitudes as input parameters. If the precision parameter is not specified, the maximum precision is used.

Example:

```
SELECT ST_GeoHash(ST_Point(-102.849854, 36.451113), 8);
SELECT ST_GeoHash(ST_GeomFromText('POINT(-102.849854 36.451113)'));
SELECT ST_GeoHash(-102.849854, 36.451113, 10);
```

1.6.12.6.2. ST_PointFromGeoHash

Function declaration:

```
st_geometry ST_PointFromGeoHash(string geohash, integer precision=full_precision)
```

Description: This function returns a point based on the input Geohash value. If the precision parameter is not specified, the maximum precision is used.

Example:

```
SELECT ST_AsText(ST_PointFromGeoHash('9wqz7eep0eyq'));
SELECT ST_AsText(ST_PointFromGeoHash('9wqz7eep0eyq', 4));
```

1.6.12.6.3. ST_EnvelopeFromGeoHash

Function declaration:

```
st_geometry ST_EnvelopeFromGeoHash(string geohash, integer precision=full_precision)
```

Description: This function returns the envelope of the specified precision based on the input Geohash value. If the precision parameter is not specified, the maximum precision is used.

Example:

```
SELECT ST_AsText(ST_EnvelopeFromGeoHash('9wqz7eep0eyq', 8));
SELECT ST_AsText(ST_EnvelopeFromGeoHash('9wqz7eep0eyq'));
```

1.6.12.6.4. ST_GeoHashNeighbours

Function declaration:

```
list_of_string ST_GeoHashNeighbours(double longitude, double latitude, integer precision)
```

Description: This function is a user-defined table-valued function (UDTF) that generates nine data records. This function returns nine Geohash strings of the current point and its eight neighboring points based on the input longitude, latitude, and precision. These parameters must be specified.

Example:

```
SELECT ST_GeoHashNeighbours(-102.849854, 36.451113, 10);
```

1.6.12.7. S2 mesh functions

1.6.12.7.1. ST_S2CellIdsFromGeom

Function declaration:

```
list_of_string ST_S2CellIdsFromGeom(st_geometry geometry, integer level)
```

Description: This function overwrites the input geometry by using S2 cells at the specified level. Then, it returns the IDs of all S2 cells.

Example:

```
SELECT ST_S2CellIdsFromGeom(ST_Point(-102.849854, 36.451113), 4);
SELECT ST_S2CellIdsFromGeom(ST_LineString('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-7
1.161144 42.25932)'), 17) as cellid;
```

1.6.12.7.2. ST_S2CellIdsFromText

Function declaration:

```
list_of_string ST_S2CellIdsFromText(string wkt, integer level)
```

Description: This function overwrites the well-known text (WKT) representation of the input geometry by using S2 cells at the specified level. Then, it returns the IDs of all S2 cells.

Example:

```
SELECT ST_S2CellIdsFromText(ST_GeomFromText('POINT(-102.849854 36.451113)'), 4);
SELECT ST_S2CellIdsFromText('LINESTRING(-71.160281 42.258729,-71.160837 42.259113,-71.161144 42.259
32)', 17) as cellid;
```

1.6.12.7.3. ST_S2CellCenterPoint

Function declaration:

```
st_point ST_S2CellCenterPoint(string cellId)
```

Description: This function calculates the center point of the cell specified by the cellId parameter in the input S2 cell.

Example:

```
SELECT ST_S2CellCenterPoint('549015');
SELECT ST_AsText(ST_S2CellCenterPoint('89e37f091'));
```

1.6.12.7.4. ST_S2CellNeighbours

Function declaration:

```
list_of_string ST_S2CellNeighbours(string cellId, integer level)
```

Description: This function calculates the neighboring S2 cells of the cell specified by the cellId parameter at the specified level. Then, it returns the IDs of all neighboring S2 cells.

Example:

```
SELECT ST_S2CellNeighbours('549015', 10);
SELECT ST_S2CellNeighbours('89e37f091', 16) as neighour;
```

1.6.12.8. Geodesic functions

1.6.12.8.1. ST_AreaWGS84

Function declaration:

```
double ST_AreaWGS84(st_geometry geometry)
```

Description: This function returns the approximate geodesic area of the input geometry based on World Geodetic System 1984 (WGS84). This function converts the coordinates of the input geometry from EPSG:4326 to EPSG:3857. Then, it calculates the plane area in square meters.

Example:

```
SELECT ST_AreaWGS84(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450, 743265 2967450, 743265.625 2967416,743238 2967416))'));
```

1.6.12.8.2. ST_DistanceWGS84

Function declaration:

```
double ST_DistanceWGS84(st_geometry geometry1, st_geometry geometry2)
```

Description: This function returns the approximate geodesic distance of the input geometry based on World Geodetic System 1984 (WGS84). This function converts the coordinates of the input geometry from EPSG:4326 to EPSG:3857. Then, it calculates the plane distance in meters.

Example:

```
SELECT ST_DistanceWGS84(ST_GeomFromText('POINT(-72.1235 42.3521)'), ST_GeomFromText('LINESTRING(-72.1260 42.45, -72.123 42.1546)'));
```

1.6.12.8.3. ST_BufferWGS84

Function declaration:

```
st_geometry ST_BufferWGS84(st_geometry geometry, double radius)
```

Description: This function returns the approximate geodesic buffer of the input geometry based on World Geodetic System 1984 (WGS84). This function converts the coordinates of the input geometry from EPSG:4326 to EPSG:3857. Then, it calculates the plane buffer and converts the coordinates back to EPSG:4326.

Example:

```
SELECT ST_AsText(ST_BufferWGS84(ST_GeomFromText('POINT(-72.1235 42.3521)'), 10));
```

1.6.12.8.4. ST_GeodesicDistance

Function declaration:

```
double ST_GeodesicDistance(double lon1, double lat1, double lon2, double lat2, string method = VINCE
NTY)
double ST_GeodesicDistance(st_geometry geo1, st_geometry geo2, string method = VINCENTY)
```

Description: This function calculates the geodesic distance between two points by using the specified method. The supported methods are Vincenty, LawOfCosines, and Haversine. The default value of the method parameter is VINCENTY. The return value is in radians.

Example:

```
SELECT ST_GeodesicDistance(ST_GeomFromText('POINT(152.352298 -24.875975)'), ST_GeomFromText('P
OINT(151.960336 -24.993289)'), 'LawOfCosines');
```

1.6.12.8.5. ST_Distance_Sphere

Function declaration:

```
double ST_Distance_Sphere(st_point geo1, st_point2 geo2)
double ST_Distance_Sphere(double lng1, double lat1, double lng2, double lat2)
```

Description: This function uses the algorithm provided by AMAP to calculate the approximate geodesic distance between the two input points. This function uses ST_Point or the specified longitudes and latitudes as input parameters.

Example:

```
SELECT ST_Distance_Sphere(
        ST_GeomFromText('POINT(116.292078 39.919622)'),
        ST_GeomFromText('POINT(116.286676 39.919593)'));
+------------+
| _c0      |
+------------+
| 460.6965312526471 |
+------------+
```

1.6.12.8.6. ST_Area_Sphere

Function declaration:

```
double ST_Area_Sphere(st_geometry geo)
```

Description: This function uses the algorithm provided by AMAP to calculate the geodesic area of the input geometry. This function uses only ST_Polygon and ST_MultiPolygon as input parameters.

Example:

```
SELECT geospatial.ST_Area_Sphere(geospatial.ST_GeomFromText('POLYGON((116.259097 40.202114,11
6.259024 40.20199,116.258768 40.201662,116.258376 40.201341,116.258031 40.201036,116.257675 40.200734
,116.257429 40.200656,116.257357 40.200562,116.257392 40.200051,116.257506 40.199433,116.257569 40.19
8586,116.257564 40.19756,116.257561 40.197372,116.257552 40.197036,116.257554 40.19675,116.257539 40.
196647,116.257502 40.19653,116.257343 40.196389,116.257153 40.196276,116.256733 40.196071,116.25582 4
0.195646,116.255628 40.195611,116.255468 40.195653,116.255385 40.195742,116.255347 40.195849,116.2552
58 40.197143,116.255103 40.199576,116.255078 40.200585,116.251227 40.20059,116.251098 40.203978,116.2
59433 40.204111,116.259247 40.203079,116.259097 40.202114))'));
+------------+
| _c0        |
+------------+
| 353493.765625 |
+------------+
```

1.6.12.9. R-tree index functions

1.6.12.9.1. ST_BuildRTreeIndex

Function declaration:

```
RTree ST_BuildRTreeIndex(string uniqueId, string geometryWkt)
```

Description: This function is a user-defined table-valued function (UDAF). It uses the unique ID and well-known text (WKT) string of each geometry as input parameters to create the R-tree index. This function must be used with other R-tree functions.

Example:

```
SELECT geospatial.ST_BuildRTreeIndex(id, shape) AS index FROM poi_sample;
```

1.6.12.9.2. ST_ContainsFromRTree

Function declaration:

```
ST_ContainsFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that are contained by the geometry from the R-tree index. This function is used to accelerate the ST_Contains query.

1.6.12.9.3. ST_CrossesFromRTree

Function declaration:

```
ST_CrossesFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that cross the geometry from the R-tree index. This function is used to accelerate the ST_Crosses query.

1.6.12.9.4. ST_EqualsFromRTree

Function declaration:

```
ST_EqualsFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that equal the geometry from the R-tree index. This function is used to accelerate the ST_Equals query.

1.6.12.9.5. ST_IntersectsFromRTree

Function declaration:

```
ST_IntersectsFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that intersect with the geometry from the R-tree index. This function is used to accelerate the ST_Intersects query.

1.6.12.9.6. ST_OverlapsFromRTree

Function declaration:

```
ST_OverlapsFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that overlap with the geometry from the R-tree index. This function is used to accelerate ST_Overlaps query.

1.6.12.9.7. ST_TouchesFromRTree

Function declaration:

```
ST_TouchesFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that spatially touch the geometry from the R-tree index. This function is used to accelerate the ST_Touches query.

1.6.12.9.8. ST_WithinFromRTree

Function declaration:

```
ST_WithinFromRTree(string uniqueId, string geometryWkt, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of objects that include the geometry from the R-tree index. This function is used to accelerate the ST_Within query.

1.6.12.9.9. ST_KNNFromRTree

Function declaration:

```
ST_KNNFromRTree(string uniqueId, string geometryWkt, int k, RTree rtree)
```

Description: This function is a user-defined table-valued function (UDTF). It uses the unique ID and well-known text (WKT) string of each geometry and the R-tree index that is created by calling ST_BuildRTreeIndex as input parameters. This function returns the IDs of k objects that are near to the geometry from the R-tree index.

1.6.12.9.10. Example

This topic provides examples on how to use R-tree index functions.

Example 1

```
-- Query the intersections of line segments in the A table and polygons in the B table.
set odps.sql.allow.cartesian=true;
SELECT a.id as link_id, b.id as shape_id
FROM link_sample_wkt a, poi_sample_wkt b
WHERE geospatial.ST_IsValid(b.shape)
AND geospatial.ST_Intersects(
    geospatial.ST_LineString(a.line),
    geospatial.ST_Multipolygon(b.shape));
Summary:
resource cost: cpu 3.28 Core * Min, memory 5.76 GB * Min
inputs:
  meta_dev.poi_sample_wkt: 1000 (237592 bytes)
  meta_dev.link_sample_wkt: 1000 (105940 bytes)
```

```
outputs:
Job run time: 111.000
+---------+----------+
| link_id | shape_id |
+---------+----------+
| 5121371185457659960 | B000A844XK |
| 5121377123249946651 | B000A85TV4 |
| 5121377166199619654 | B000A844KT |
+---------+----------+
-- After optimization by using the new function:
SELECT /*+mapjoin(i)*/
    geospatial.ST_IntersectsFromRTree(id, line, i.index)
    AS (link_id, shape_id)
FROM link_sample_wkt
JOIN
(
    SELECT geospatial.ST_BuildRTreeIndex(id, shape) AS index
    FROM poi_sample_wkt
    WHERE geospatial.ST_IsValid(shape)
) i;
Summary:
resource cost: cpu 1.03 Core * Min, memory 1.99 GB * Min
inputs:
  meta_dev.poi_sample_wkt: 1000 (237592 bytes)
  meta_dev.link_sample_wkt: 1000 (105940 bytes)
outputs:
Job run time: 41.000
+---------+----------+
| link_id | shape_id |
+---------+----------+
| 5121371185457659960 | B000A844XK |
| 5121377123249946651 | B000A85TV4 |
| 5121377166199619654 | B000A844KT |
+---------+----------+
```

Example 2

```
-- Create an R-tree for all points in a table and use the KNN function to locate the nearest point of eac
h point.
SELECT /*+mapjoin(i)*/
   geospatial.ST_KNNFromRTree(id, point, 1, i.index) AS (id1, id2)
FROM poi_sample_wkt
JOIN
(
   SELECT geospatial.ST_BuildRTreeIndex(id, point) AS index
   FROM poi_sample_wkt
) i;
Summary:
resource cost: cpu 1.17 Core * Min, memory 2.24 GB * Min
inputs:
  meta_dev.poi_sample_wkt: 1000 (237592 bytes)
outputs:
Job run time: 46.000
+-----+-----+
| id1 | id2 |
+-----+-----+
| B000A01B4E | B000A01B4E |
| B000A01C19 | B000A01C19 |
| B000A023A5 | B000A023A5 |
| B000A02F81 | B000A02F81 |
| B000A07BEE | B000A07BEE |
| B000A07E06 | B000A07E06 |
| B000A08863 | B000A08863 |
...
-- The table has 1,000 rows of data. This function returns 1,000 rows of data, which meets your expect
ations.
```

1.6.12.10. Other functions

1.6.12.10.1. ST_IsValid

Function declaration:

```
boolean ST_IsValid(st_geometry geometry)
boolean ST_IsValid(string wkt)
```

Description: This function checks whether the input geometry or well-known text (WKT) string meets the requirements.

Example:

```
SELECT ST_IsValid('POINT(-102.849854 36.451113)');
SELECT ST_IsValid(ST_Point('POINT(-102.849854 36.451113)'));
```

1.6.12.10.2. ST_Transform

Function declaration:

```
st_geometry ST_TransformWGS84(st_geometry geometry)
st_geometry ST_Transform(st_geometry geometry, integer toSRID)
st_geometry ST_Transform(st_geometry geometry, integer fromSRID, integer toSRID)
```

Description: This function converts the coordinates of the input geometry from one spatial reference system to another. The ST_TransformWGS84 function converts the coordinates of the geometry from EPSG:4326 to EPSG:3857. The ST_Transform function converts the geometry from fromSRID to toSRID. If the overload function contains only toSRID, you must call the ST_SetSRID function first.

Example:

```
SELECT ST_AsText(ST_Transform(ST_GeomFromText('POLYGON((743238 2967416,743238 2967450, 74326
5 2967450,743265.625 2967416,743238 2967416))', 2249, 4326));
SELECT ST_AsText(ST_TransformWGS84(ST_GeomFromText('POLYGON((-71.1776848522251 42.39028965
12902,-71.1776843766326 42.3903829478009, -71.1775844305465 42.3903826677917,-71.1775825927231 42.
3902893647987,-71.1776848522251 42.3902896512902))')));
```

1.6.13. SQL Function

User-defined functions (UDFs) in MaxCompute support Java or Python. Some UDFs can be directly implemented by SQL. Therefore, MaxCompute supports SQL functions. This improves the reuse rate of SQL code.

Use SQL functions

Example:

```
FUNCTION ADD(@a BIGINT) AS @a + 1;
SELECT ADD(key), ADD(value) FROM src;
```

Functions as input parameters

Functions can be used as input parameters for SQL functions, including built-in functions, UDFs, and SQL functions.

Example:

```
FUNCTION ADD(@a BIGINT) AS  @a + 1;
FUNCTION OP(@a,  @fun FUNCTION (BIGINT) RETURNS BIGINT) AS  @ fun(@a);
SELECT OP(key, ADD), OP(key, abs) FROM src;
```

Anonymous functions as input parameters

Anonymous functions can be used as input parameters for SQL functions.

Example:

```
FUNCTION OP(@a,  @fun FUNCTION (BIGINT) RETURNS BIGINT) AS   @ fun(@a);
SELECT OP(key, FUNCTION (@a) AS  @a + 1) FROM src;
```

1.6.14. CLONE TABLE

MaxCompute supports the CLONE TABLE statement. You can execute this statement to clone data from one table to another.

Syntax

```
CLONE TABLE <[src_project_name.]src_table_name> [PARTITION(spec), ...]  TO <[dest_project_name.]de
sc_table_name> [IF EXISTS (OVERWRITE | IGNORE)] ;
```

> **Note**
>
> - If the destination table is not created before data is cloned, a table is created by using the CREATE TABLE LIKE statement when you execute the CLONE TABLE statement.
> - If the destination table is created before data is cloned and IF EXISTS OVERWRITE is specified, data in the specified partitions of the destination table is overwritten.
> - If the destination table is created before data is cloned and IF EXISTS IGNORE is specified, existing partitions in the destination table are skipped and data in these partitions is not overwritten.

Limits and troubleshooting

- The schema of a destination table must be compatible with that of the source table.
- The CLONE TABLE statement supports both partitioned and non-partitioned tables. Tables that have special data organization structures are not supported. These tables include clustered tables, shard tables, Xlib or Algo tables, and tables with extreme storage.
- Make sure that the configuration of the cluster for the source table intersects with that for the destination table and the data that you want to process is in the same cluster. If any of the conditions is not met, an error is returned.
- If the destination table already exists before data is cloned, you can clone data from a maximum of 10,000 partitions at a time.
- If the destination table does not exist before data is cloned, the number of partitions that you can clone at a time is not limited, which ensures atomicity.

- If a hard link in the Apsara Distributed File System is faulty, purge the recycle bin and try again.
- The user who submits the command must have the Create Table and Update Table permissions on the target project.

Example

The following code shows the partitions and data of the source tables:

```
odps@ multi>read srcpart_copy;
+------------+------------+------------+------------+
| key      | value     | ds        | hr        |
+------------+------------+------------+------------+
| 1        | ok49      | 2008-04-09 | 11        |
| 1        | ok48      | 2008-04-08 | 12        |
+------------+------------+------------+------------+
odps@ multi>read src_copy;
+------------+------------+
| key      | value     |
+------------+------------+
| 1        | ok        |
+------------+------------+
```

Clone all data from the non-partitioned table.

```
clone table src_copy to src_clone;
odps@ multi>clone table src_copy to src_clone;
ID = 20191023030245544g2540cdv2
OK
odps@ multi>read src_clone;
+------------+------------+
| key      | value     |
+------------+------------+
| 1        | ok        |
+------------+------------+
```

Clone some partitions of the partitioned table.

```
clone table srcpart_copy partition(ds="2008-04-09", hr='11') to srcpart_clone IF EXISTS OVERWRITE;
odps@ multi>clone table srcpart_copy partition(ds="2008-04-09", hr='11') to srcpart_clone IF EXISTS OV
ERWRITE;
ID = 20191023030534986g4540cdv2
OK
odps@ multi>read srcpart_clone;
+------------+------------+------------+------------+
| key      | value     | ds       | hr       |
+------------+------------+------------+------------+
| 1        | ok49      | 2008-04-09 | 11       |
+------------+------------+------------+------------+
```

Clone data from the partitioned table and skip existing partitions in the destination table.

```
clone table srcpart_copy to srcpart_clone IF EXISTS IGNORE;
odps@ multi>clone table srcpart_copy to srcpart_clone IF EXISTS IGNORE;
ID = 20191023030619196g5540cdv2
OK
odps@ multi>read srcpart_clone;
+------------+------------+------------+------------+
| key      | value     | ds       | hr       |
+------------+------------+------------+------------+
| 1        | ok49      | 2008-04-09 | 11       |
| 1        | ok48      | 2008-04-08 | 12       |
+------------+------------+------------+------------+
```

Clone all data from the partitioned table.

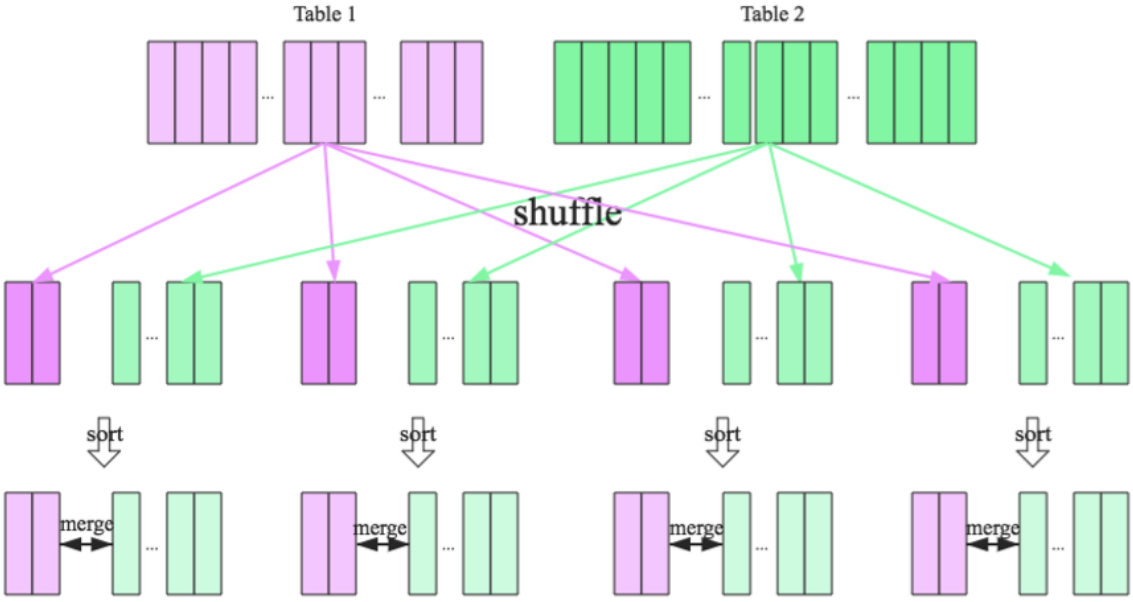
```
clone table srcpart_copy to srcpart_clone2;
odps@ multi>clone table srcpart_copy to srcpart_clone2;
ID = 20191023030825186g6540cdv2
OK
odps@ multi>read srcpart_clone2;
+------------+------------+------------+------------+
| key      | value     | ds       | hr       |
+------------+------------+------------+------------+
| 1        | ok49      | 2008-04-09 | 11       |
| 1        | ok48      | 2008-04-08 | 12       |
+------------+------------+------------+------------+
```

1.6.15. MaxCompute Hash Clustering

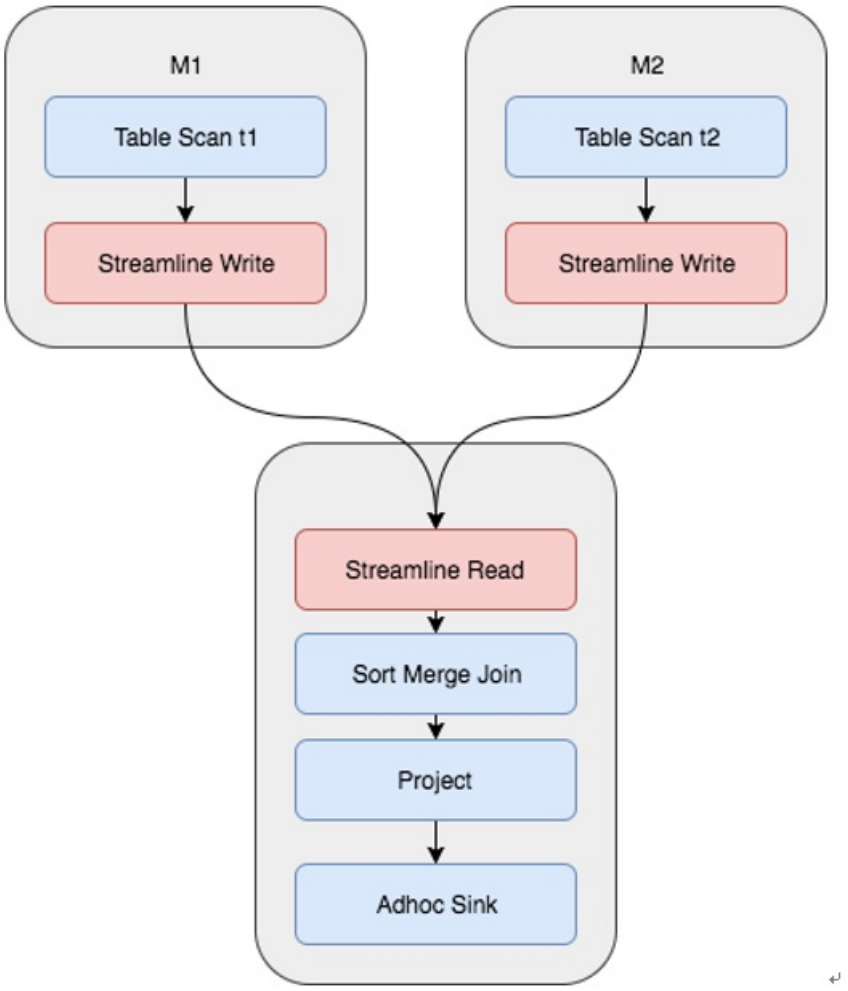
1.6.15.1. Background information

JOIN operations are commonly used for queries in MaxCompute. MaxCompute provides the following implementation methods of JOIN operations:

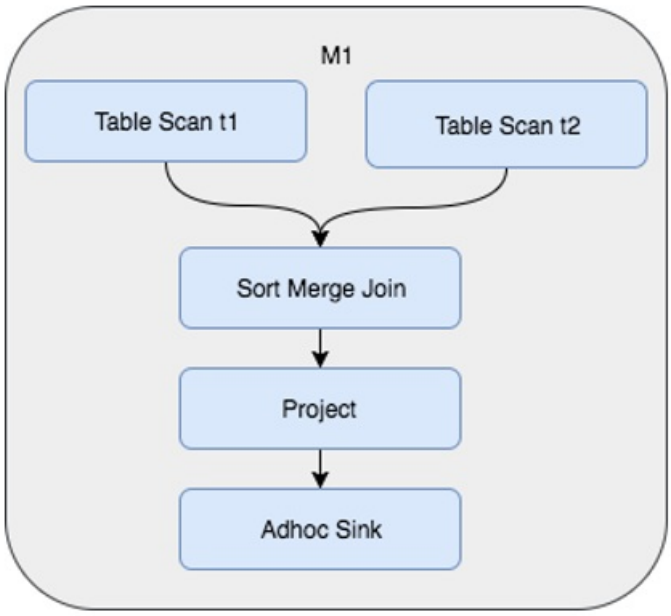
1. **Broadcast hash join**: This method is used when a JOIN operation involves a small table. The small table is broadcasted and transferred to all JoinTask instances. Then, the hash join operation is performed to join the small table with a large table.
2. **Shuffle hash join**: This method is used when a JOIN operation involves large tables that cannot be broadcasted directly. In this case, the hash shuffle operation is performed on two tables based on join keys. The hash results for the same key-value pairs are the same. This ensures that results that have the same key are collected on a JoinTask instance. For each instance, a hash table is created by using a small table, probe operations are performed by using a large table, and then the tables are joined.
3. **Sort merge join**: This method is used when a JOIN operation involves larger tables and the preceding methods cannot be used because the memory is insufficient to create a hash table. In this case, the hash shuffle operation is performed on two tables based on join keys, the obtained values are sorted by using join keys, and then the sorted values are merged.



The **sort merge join** operation is commonly used in MaxCompute because MaxCompute processes huge volumes of data in most cases. This operation generates repeated shuffle and join operations. The physical execution plan of Job Scheduler of the JOIN operation also requires multiple stages, which consumes excessive volumes of resources.



Therefore, MaxCompute allows you to configure the hash shuffle and sort attributes when the data is initially generated in a table. This prevents data from being shuffled and sorted repeatedly in subsequent queries. As a result, the number of stages in the physical execution plan of Job Scheduler of a JOIN operation is reduced. The preceding figure shows that only one stage is required.



MaxCompute Hash Clustering allows you to configure the shuffle and sort attributes of a table when you create the table. As a result, MaxCompute optimizes the execution plan, improves the efficiency, and saves resources based on the existing storage characteristics.

1.6.15.2. Descriptions

1.6.15.2.1. Enable or disable Hash Clustering

The Hash Clustering feature is available and enabled by default. If you want to use clustered indexes, add the following flag:

```
set odps.sql.cfile2.enable.read.write.index.flag=true;
```

After the flag is set to true, the system automatically creates indexes for the sorted hash buckets to improve query efficiency. To use clustered indexes, you must add this flag during table creation and subsequent queries. If you want to use clustered indexes in your project all the time, contact the MaxCompute team.

> **Note** Clustered indexes improve the efficiency of queries (equivalent values or ranges) based on sort keys. However, you can still experience the superior performance provided by Hash Clustering although you do not add this flag.

1.6.15.2.2. Create a hash clustering table

You can use the following statement to create a hash clustering table. You must specify cluster keys or hash keys and the number of hash buckets. The sort operation is optional. However, we recommend that you use the same parameter values as the cluster keys to achieve optimal performance.

```
CREATE TABLE [IF NOT EXISTS] table_name
  [(col_name data_type [comment col_comment], ...)]
  [comment table_comment]
  [PARTITIONED BY (col_name data_type [comment col_comment], ...)]
[CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [, col_name [ASC | DESC] ...])] INTO number_of_buckets BUCKETS]
[AS select_statement]
```

You can use the following statement to create a standard table:

```
CREATE TABLE T1 (a string, b string, c bigint) CLUSTERED BY (c) SORTED by (c) INTO 1024 BUCKETS;
```

You can use the following statement to create a partitioned table:

```
CREATE TABLE T1 (a string, b string, c bigint) PARTITIONED BY (dt string) CLUSTERED BY (c) SORTED by (c) INTO 1024 BUCKETS;
```

The following sections detail the CLUSTERED BY, SORTED BY, and INTO number_of_buckets BUCKETS clauses.

CLUSTERED BY

The **CLUSTERED BY** clause specifies hash keys. MaxCompute performs the hash operation on the specified column and distributes data to buckets based on the hash values. To prevent data skew and hot spots, and to concurrently execute statements, we recommend that you specify a column that has large value ranges and a small number of duplicate key-value pairs in CLUSTERED BY. In addition, to optimize the JOIN operation, we recommend that you select commonly used join or aggregation keys. The join and aggregation keys are similar to the primary keys in conventional databases.

SORTED BY

The **SORTED BY** clause specifies how fields are sorted in a bucket. We recommend that you specify the same column in SORTED BY as that in CLUSTERED BY to improve execution efficiency. After you specify the column in SORTED BY, MaxCompute automatically generates indexes and then executes SQL statements faster when you query data based on these indexes.

INTO number_of_buckets BUCKETS

The **INTO number_of_buckets BUCKETS** clause specifies the number of hash buckets, which is required. The number of hash buckets is determined by the volume of data. More buckets indicate higher concurrency, which shortens the job running time. However, if too many buckets exist, excessive small files may be generated. In addition, high concurrency increases CPU time. We recommend that you set the volume of data for each bucket to a value that ranges from 500 MB to 1 GB. If a large table is used, you can adjust the value to a larger value as required.

You can remove the shuffle operation only for tables with the same number of buckets in MaxCompute. In later versions, MaxCompute will support bucket alignment. You will be able to remove the shuffle operation for tables whose numbers of buckets are multiples or factors of each other. To achieve bucket alignment, we recommend that you set the number of buckets to a power of 2, for example, 512, 1,024, and 2,048. The maximum number of buckets is 4,096. If the number of buckets exceeds the value, the performance and resource usage may be affected.

If you want to remove the shuffle and sort operations during a JOIN operation on two tables, the numbers of hash buckets in the tables must be the same. If the numbers that are calculated based on the aforementioned method are inconsistent, we recommend that you use the larger number for the JOIN operation. This guarantees that SQL statements can be executed concurrently in an efficient manner.

If the sizes of two tables greatly differ, you can set the number of buckets for the large table to several times of that for the small table, for example, 256 and 1,024. If automatic hash bucket split and merging are supported, the settings can be optimized by using data features.

1.6.15.2.3. Modify table attributes

For a partitioned table, MaxCompute allows you to execute the ALTER TABLE statement to add the Hash Clustering attribute to a table or remove the Hash Clustering attribute from a table.

```
ALTER TABLE table_name
  [CLUSTERED BY (col_name [, col_name, ...]) [SORTED BY (col_name [ASC | DESC] [, col_name [ASC | DES
C] ...])] INTO number_of_buckets BUCKETS]
ALTER TABLE table_name NOT CLUSTERED;
```

Note the following points when you use the ALTER TABLE statement:

- The ALTER TABLE statement can only modify the Hash Clustering attribute of a partitioned table. The Hash Clustering attribute cannot be modified after it is added to a non-partitioned table.
- The ALTER TABLE statement takes effect only for the new partitions of a table, which include the partitions generated by using the INSERT OVERWRITE statement. New partitions are stored based on the Hash Clustering attribute. The storage formats of existing partitions remain unchanged.
- The ALTER TABLE statement takes effect only for the new partitions of a table. Therefore, you cannot specify a partition in this statement.

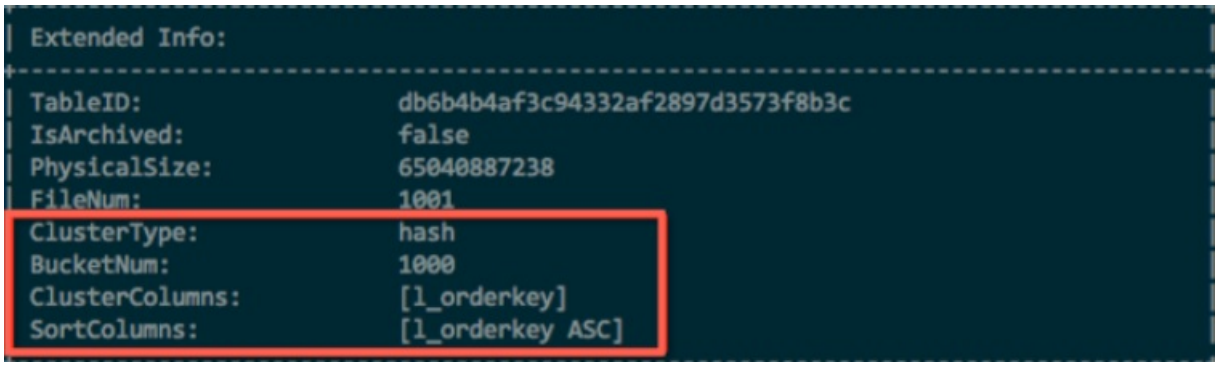
The ALTER TABLE statement is suitable for existing tables. After the Hash Clustering attribute is added, new partitions are stored based on the Hash Clustering attribute.

1.6.15.2.4. View and verify table attributes

After you create a hash clustering table, execute the following statement to view table attributes:

```
DESC EXTENDED table_name;
```

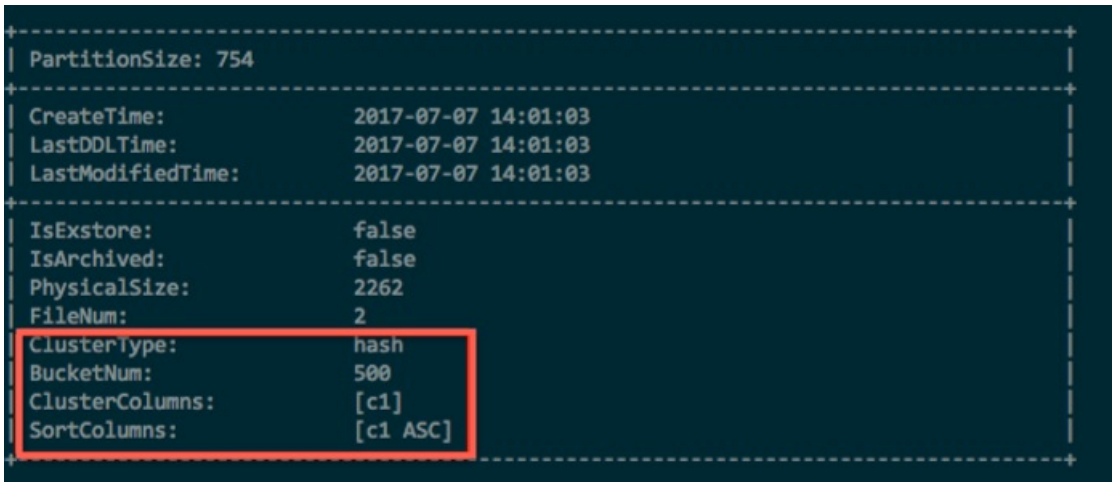
The Hash Clustering attribute is displayed in Extended Info.



You can also execute the following statement to view partition attributes of a partitioned table:

```
DESC EXTENDED table_name partition(pt_spec);
```

The following figure shows the execution result.



1.6.15.3. Benefits

1.6.15.3.1. Bucket pruning and index optimization

The following code provides a syntax sample:

```
CREATE TABLE t1 (id bigint, a string, b string) CLUSTERED BY (id) SORTED BY (id) into 1000 BUCKETS;
...
SELECT t1.a, t1.b, t1.c FROM t1 WHERE t1.id=12345;
```

This syntax indicates a full scan for a standard table. A full scan for a large table consumes a large number of resources. However, if the hash shuffle operation is performed on all id fields and the id fields are sorted, the query is greatly simplified. The sample procedure is as follows:

1. Find the hash bucket that corresponds to 12345. This query is performed in only one bucket, not all 1,000 buckets. This process is called bucket pruning.
2. Data in a bucket is stored based on IDs. MaxCompute automatically creates indexes and uses the INDEX LOOKUP function to locate relevant records.

The simplified procedure not only greatly reduces the number of mappers, but also allows mappers to locate the page where the data is stored by using the INDEX function. Therefore, the volume of loaded data is greatly reduced.

1.6.15.3.2. Aggregation optimization

The following code provides a syntax sample:

```
SELECT department, SUM(salary) FROM employee GROUP BY (department);
```

In most cases, the department column is shuffled and sorted. Then, a stream aggregate operation is performed to collect statistics on the department groups. However, if `CLUSTERED BY (department) SORTED BY (department)` is executed for the table data, the shuffle and sort operations are no longer required.

1.6.15.3.3. Storage optimization

In addition to computation optimization, storage space is greatly saved if tables are shuffled and stored in a sorted manner. MaxCompute uses the column store at the underlying layer. Records with the same or similar key-value pairs are stored together by the sort function, which facilitates encoding and compression. As a result, compression efficiency is improved. In some cases, a sorted table can save 50% more storage space than an unsorted table. Therefore, Hash Clustering is suitable for the storage of tables that have long lifecycles.

For example, take a table with 100 GB of TPC-H line items and multiple data types, such as INT, DOUBLE, and STRING. When Hash Clustering is used, about 10% of the storage space is saved while the volume of data and compression format remain unchanged.

```
CreateTime:                2016-04-17 21:48:08
LastDDLTime:               2016-04-17 21:48:08
LastModifiedTime:          2016-04-17 21:50:10
InternalTable: YES         Size: 23573055432
Native Columns:
Field             Type        Label  Comment
l_orderkey        bigint
l_partkey         bigint
l_suppkey         bigint
l_linenumber      bigint
l_quantity        double
l_extendedprice   double
l_discount        double
l_tax             double
l_returnflag      string
l_linestatus      string
l_shipdate        string
l_commitdate      string
l_receiptdate     string
l_shipinstruct    string
l_shipmode        string
l_comment         string
```

```
CreateTime:                2017-07-13 14:40:11
LastDDLTime:               2017-07-13 14:40:11
LastModifiedTime:          2017-07-13 15:05:04
InternalTable: YES         Size: 21658913950
Native Columns:
Field             Type        Label  Comment
l_orderkey        bigint
l_partkey         bigint
l_suppkey         bigint
l_linenumber      bigint
l_quantity        double
l_extendedprice   double
l_discount        double
l_tax             double
l_returnflag      string
l_linestatus      string
l_shipdate        string
l_commitdate      string
l_receiptdate     string
l_shipinstruct    string
l_shipmode        string
l_comment         string
```

1.6.15.4. ShuffleRemove

- Range clustering tables support the join and aggregate operations. If a join or group key is a range clustering key or its prefix, data redistribution is not required. This mechanism is called ShuffleRemove, which improves execution efficiency.

  Usage: The `odps.optimizer.enable.range.partial.repartitioning` flag controls whether to enable this feature. This feature is disabled by default.

- If you join two hash clustering tables and the numbers of buckets in these tables are different but are multiples or factors of each other, data redistribution is not required. This improves execution efficiency.

  Usage: The `odps.optimizer.enable.hash.partial.repartitioning` flag controls whether to enable this feature. This feature is enabled by default.

- Correlated Shuffle Remove is supported. If data meets distribution requirements but does not meet the sorting requirements, you can add a sort operator to avoid data redistribution.

1.6.15.5. Limits

The limits of Hash Clustering are described as follows:

- The INSERT INTO statement is not supported. You can only execute the INSERT OVERWRITE statement to import data.
- Small files cannot be merged. Data is evenly distributed in buckets when it is split, so no small files are generated. If you merge files, the data distribution is affected. However, you can still use the merge and archive commands to change the storage format of a table file and the format of a RAID file.
- You cannot use Tunnel to upload data to a range-clustered table because data uploaded by using Tunnel is unsorted.

In the future, these limits will be resolved. Stay tuned for updates on the official website.

1.6.16. MaxCompute SQL limits

The following table lists all the limits of MaxCompute SQL statements.

Limits

Item	Maximum value/Limit	Category	Description
Table name length	128 bytes	Length	A table name or column name cannot contain special characters. It can contain only lowercase and uppercase letters, digits, and underscores (_) and must start with a letter.
Comment length	1,024 bytes	Length	A comment can be up to 1,024 bytes in length.
Column definitions in a table	1,200	Quantity	A table can contain a maximum of 1,200 column definitions.

Item	Maximum value/Limit	Category	Description
Partitions in a table	60,000	Quantity	A table can contain a maximum of 60,000 partitions.
Partition levels of a table	6	Quantity	A table can contain a maximum of six levels of partitions.
Statistical definitions of a table	100	Quantity	A table can contain a maximum of 100 statistical definitions.
Statistical definition length of a table	64,000	Length	The length of statistical definitions in a table cannot exceed 64,000.
Screen display	10,000 rows	Quantity	A SELECT statement can generate a maximum of 10,000 rows.
INSERT targets	256	Quantity	A MULTIINS operation can insert a maximum of 256 data tables at a time.
UNION ALL	256 tables	Quantity	A UNION ALL operation can be performed on a maximum of 256 tables.
JOIN sources	128	Quantity	The JOIN operation can be performed on a maximum of 128 source tables.
MAPJOIN memory	512 MB	Quantity	The memory size for all small tables on which the MAPJOIN operation is performed cannot exceed 512 MB.
Window functions	5	Quantity	A SELECT statement can contain a maximum of five window functions.
PTINSUBQ	1,000 rows	Quantity	A PT IN SUBQUERY statement can generate a maximum of 1,000 rows.
Length of an SQL statement	2 MB	Length	The maximum length of an SQL statement is 2 MB.
Conditions of a WHERE clause	256	Quantity	A WHERE clause can contain a maximum of 256 conditions.
Length of a column record	8 MB	Length	The maximum length of a column record in a table is 8 MB.
IN parameters	1,024	Quantity	This item specifies the maximum number of parameters in an IN clause, such as in(1,2,3,...,1024). Excess parameters can slow down the compilation process. We recommend that you use no more than 1,024 parameters, but this is not a fixed upper limit.

Item	Maximum value/Limit	Category	Description
jobconf.json	1 MB	Length	The maximum size of the jobconf.json file is 1 MB. If a table contains a large number of partitions, the size of jobconf.json may exceed 1 MB.
View	Not writable	Operation	A view is not writable and does not support the INSERT operation.
Data type and position of a column	Unmodifiable	Operation	The data type and position of a column are unmodifiable.
Java UDFs	Cannot be abstract or static	Operation	Java UDFs cannot be abstract or static.
Partitions to query	10,000	Quantity	A maximum of 10,000 partitions can be queried.

Notice The preceding MaxCompute SQL limits cannot be modified manually.

1.6.17. Common MaxCompute SQL parameter settings

1.6.17.1. MAP configurations

```
set odps.sql.mapper.cpu=100
```

Purpose: It is used to set the number of CPUs for each instance in a Map task. Default value: 100. Value range: 50 to 800.

```
set odps.sql.mapper.memory=1024
```

Purpose: It is used to set the memory size for each instance in a Map task. Default value: 1024 MB. Value range: 256 MB to 12,288 MB.

```
set odps.sql.mapper.merge.limit.size=64
```

Purpose: It is used to set the maximum size of control files to be merged. Default value: 64 MB. You can set this variable to control the inputs of mappers. Value range: 0 to Integer.MAX_VALUE.

```
set odps.sql.mapper.split.size=256
```

Purpose: It is used to set the maximum data input volume for a map. Default value: 256 MB. You can set this variable to control the inputs of mappers. Value range: 1 to Integer.MAX_VALUE.

1.6.17.2. JOIN configurations

```
set odps.sql.joiner.instances=-1
```

Purpose: It is used to set the number of instances in a JOIN task. Default value: 1. Value range: 0 to 2,000.

```
set odps.sql.joiner.cpu=100
```

Purpose: It is used to set the number of CPUs for each instance in a JOIN task. Default value: 100. Value range: 50 to 800.

```
set odps.sql.joiner.memory=1024
```

Purpose: It is used to set the memory size for each instance in a JOIN task. Default value: 1,024 MB. Value range: 256 MB to 12,288 MB.

1.6.17.3. Reduce configurations

```
set odps.sql.reducer.instances=-1
```

Purpose: It is used to set the number of instances in a Reduce task. Default value: 1. Value range: 0 to 2,000.

```
set odps.sql.reducer.cpu=100
```

Purpose: It is used to set the number of CPUs for each instance in a Reduce task. Default value: 100. Value range: 50 to 800.

```
set odps.sql.reducer.memory=1024
```

Purpose: It is used to set the memory size for each instance in a Reduce task. Default value: 1,024 MB. Value range: 256 to 12,288 MB.

1.6.17.4. UDF configurations

```
set odps.sql.udf.jvm.memory=1024
```

Purpose: It is used to set the maximum memory size for a UDF JVM heap. Default value: 1,024 MB. Value range: 256 to 12,288 MB.

```
set odps.sql.udf.timeout=600
```

Purpose: It is used to set the timeout value of a UDF. Default value: 600 seconds. Value range: 0 to 3,600 seconds.

```
set odps.sql.udf.python.memory=256
```

Purpose: It is used to set the maximum memory size for UDF python. Default value: 256 MB. Value range: 64 to 3,072 MB.

```
set odps.sql.udf.optimize.reuse=true/false
```

Purpose: after start-up, each UDF function expression can only be calculated once, improving performance. The default is true.

```
set odps.sql.udf.strict.mode=false/true
```

Purpose: It is used to control functions regarding whether to return NULL or error if dirty data is encountered. If it is true, an error is returned. If it is false, NULL is returned.

1.6.17.5. MAPJOIN configurations

```
set odps.sql.mapjoin.memory.max=512
```

Purpose: It is used to set the maximum memory of a small table in MAPJOIN. Default vlaue 512 MB. Value range: 128 to 2,048 MB.

```
set odps.sql.reshuffle.dynamicpt=true/false
```

Purpose:

- Some scenarios of dynamic partitioning are time-consuming. Shutting them down can speed up SQL.
- If the dynamic partition value is very small, disabling dynamic partition can avoid data skew.

1.6.17.6. Configure data skew

```
set odps.sql.groupby.skewindata=true/false
```

Effect: enables the group by optimization.

```
set odps.sql.skewjoin=true/false
```

Effect: enables the join optimization. It takes effect only when odps.sql.skewinfo is configured.

```
set odps.sql.skewinfo
```

Purpose: It is used to set detailed information of join optimization. The command syntax is as follows:

```
set odps.sql.skewinfo=skewed_src:(skewed_key)[("skewed_value")]
```

Example:

The following command is used to set a single skewed data value in a single field:

```
set odps.sql.skewinfo=src_skewjoin1:(key)[("0")]
-- Command output: explain select a.key c1, a.value c2, b.key c3, b.value c4 from src a join src_skewjoin
1 b on a.key = b.key;
```

The following command is used to set multiple skewed data values in a single field:

```
set odps.sql.skewinfo=src_skewjoin1:(key)[("0")("1")]
-- Command output: explain select a.key c1, a.value c2, b.key c3, b.value c4 from src a join src_skewjoin
1 b on a.key = b.key;
```

1.6.18. MapReduce-to-SQL conversion for execution

1.6.18.1. Overview

MaxCompute provides a series of Java APIs for MapReduce to process data.

In the current version, MapReduce programs are automatically converted to SQL for execution. After the conversion, you can use the compiler, cost-based optimizer, and vectorized execution engine released with MaxCompute V2.0 to process the MapReduce programs. The new features of the SQL engine can also be used. The features, performance, and stability of the SQL engine are optimized.

Notice

- You do not need to change the original APIs and job logic.
- Only MapReduce jobs of the OpenMR type, which are written with MapReduce APIs, can be converted to SQL.
- This feature can be used for projects and jobs.
- This feature supports views as the input.
- This feature supports external tables as the input.
- This feature supports TemporaryFile reads and writes.
- This feature allows you to read data from and write data to hash clustering tables.
- This feature supports the near-real-time execution of small jobs.

1.6.18.2. Configure local running settings

1. Download the latest MaxCompute client package to your computer and properly configure the client.
2. Configure the execution mode.

   You can configure the execution mode based on your business requirements. The default execution mode is lot. In lot mode, jobs are executed by MapReduce. The new compiler, optimizer, and execution engine are not required.

   You can enable the execution mode by setting the *odps.mr.run.mode* parameter. Valid values: **lot, sql, and hybrid**.

   - Method 1: Enable the execution mode at the project level. When the execution mode is enabled, it affects all jobs. Therefore, the project administrator must apply for and enable the execution mode. Set the *odps.mr.run.mode* parameter to **hybrid** or **sql**. If SQL execution fails in hybrid mode, the job is executed by MapReduce. If SQL execution fails in sql mode, an error is returned.
   - Method 2: Enable the execution mode at the session level. This method is only valid for the current job. To enable the execution mode, use one of the following methods:
     - Add a set flag, such as `set odps.mr.run.mode=hybrid`, before JAR statements.
     - Configure the job parameters. Example:

     ```
     JobConf job = new JobConf();
     job.set("odps.mr.run.mode","hybrid")
     ```

   The execution mode can be enabled at the project level later by MaxCompute O&M personnel.

1.6.18.3. Operation settings in DataWorks

Jobs running in DataWorks are updated by the O&M personnel of MaxCompute and DataWorks. You do not need to update the client manually.

1. Enable the conversion for a single job.

   You can add the SET statement before a MapReduce job or configure the job parameter for it. These methods take effect at the session level and apply only to the current job.

   The following examples demonstrate how to use these methods:

   - Add the SET statement, such as `set odps.mr.run.mode=hybrid`.
   - Configure the job parameter as follows:

   ```
   JobConf job = new JobConf();
   job.set("odps.mr.run.mode","hybrid")
   ```

2. Enable the conversion at the project level by setting *odps.mr.run.mode* for a project.

1.6.18.4. View running details

You can use Logview and MaxCompute Studio to view MapReduce-to-SQL conversion results and running details of SQL jobs.

1. LogView XML.

   Open Logview and click the LOT node in the center of the page. The SQL jobs that are converted from MapReduce jobs are included in the XML information of the node. Example:

```
create temporary function mr2sql_mapper_152955927079392291755 as   'com.aliyun.odps.mapred.br
idge.LotMapperUDTF' using ;
create temporary function mr2sql_reducer_152955927079392291755 as 'com.aliyun.odps.mapred.bri
dge.LotReducerUDTF' using ;
 @sub_query_mapper :=
 SELECT k_id,v_gmt_create,v_gmt_modified,v_product_id,v_admin_seq,v_sku_attr,v_sku_price,v_sku
_stock,v_sku_code,v_sku_image,v_delivery_time,v_sku_bulk_order,v_sku_bulk_discount,v_sku_imag
e_version,v_currency_code
FROM(
SELECT mr2sql_mapper_152955927079392291755(id,gmt_create,gmt_modified,product_id,admin_seq
,sku_attr,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk_order,sku_bulk_discoun
t,sku_image_version,currency_code ) as (k_id,v_gmt_create,v_gmt_modified,v_product_id,v_admin_
seq,v_sku_attr,v_sku_price,v_sku_stock,v_sku_code,v_sku_image,v_delivery_time,v_sku_bulk_order
,v_sku_bulk_discount,v_sku_image_version,v_currency_code)
FROM ae_antispam.product_sku_tt_inc
WHERE ds = "20180615"  AND hh = "21"
UNION ALL
SELECT mr2sql_mapper_152955927079392291755(id,gmt_create,gmt_modified,product_id,admin_seq
,sku_attr,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk_order,sku_bulk_discoun
t,sku_image_version,currency_code ) as (k_id,v_gmt_create,v_gmt_modified,v_product_id,v_admin_
seq,v_sku_attr,v_sku_price,v_sku_stock,v_sku_code,v_sku_image,v_delivery_time,v_sku_bulk_order
,v_sku_bulk_discount,v_sku_image_version,v_currency_code)
FROM ae_antispam.product_sku
) open_mr_alias1
 DISTRIBUTE BY k_id SORT BY k_id ASC;
@sub_query_reducer :=
SELECT mr2sql_reducer_152955927079392291755(k_id,v_gmt_create,v_gmt_modified,v_product_id,v_
admin_seq,v_sku_attr,v_sku_price,v_sku_stock,v_sku_code,v_sku_image,v_delivery_time,v_sku_bul
k_order,v_sku_bulk_discount,v_sku_image_version,v_currency_code) as (id,gmt_create,gmt_modifi
ed,product_id,admin_seq,sku_attr,sku_price,sku_stock,sku_code,sku_image,delivery_time,sku_bulk
_order,sku_bulk_discount,sku_image_version,currency_code)
FROM @sub_query_mapper;
FROM @sub_query_reducer
INSERT OVERWRITE TABLE ae_antispam.product_sku
SELECT id,gmt_create,gmt_modified,product_id,admin_seq,sku_attr,sku_price,sku_stock,sku_code,s
ku_image,delivery_time,sku_bulk_order,sku_bulk_discount,sku_image_version,currency_code ;
```

2. LogView detail or summary.

   You can see that the new execution engine is used to execute jobs.

```
Job run mode: fuxi job
 Job run engine: execution engine
```

3. LogView detail or JSON summary.

   The JSON summary information in MapReduce only contains the input and output information of Map and Reduce. However, the JSON summary information in SQL allows you to view details about each stage of SQL execution, such as all execution parameters, logical execution plans, physical execution plans, and execution details. Example:

```
"midlots" :
[
"LogicalTableSink(table=[[odps_flighting.flt_20180621104445_step1_ad_quality_tech_qp_algo_antifake_wordbag_filter_bag_change_result_lv2_20, auctionid,word,match_word(3) {0, 1, 2}]])
OdpsLogicalProject(auctionid=[$0], word=[$1], match_word=[$2])
OdpsLogicalProject(auctionid=[$0], word=[$1], match_word=[$2])
OdpsLogicalProject(auctionid=[$0], word=[$1], match_word=[$2])
OdpsLogicalProject(auctionid=[$2], word=[$3], match_word=[$4])
OdpsLogicalTableFunctionScan(invocation=[[MR2SQL_MAPPER_152955294118813063732($0, $1)]()], rowType=[RecordType(VARCHAR(2147483647) item_id, VARCHAR(2147483647) text, VARCHAR(2147483647) __tf_0_0, VARCHAR(2147483647) __tf_0_1, VARCHAR(2147483647) __tf_0_2)])
OdpsLogicalTableScan(table=[[ad_quality_tech.qp_algo_antifake_wordbag_filter_bag_change_lv2_20, item_id,text(2) {0, 1}]])
]
```

1.6.18.5. Perform operations on the distributed file system

Procedure

1. Specify volume files.

   You can use either of the following methods to specify volume files:

   - Use a utility class to specify the input and output files:

     ```
     com.aliyun. ODPS .mapred.utils.InputUtils.addVolume( new VolumeInfo([project,]inVolume,inPartition, "inLabel"), new JobConf());
     com.aliyun. ODPS .mapred.utils.OutputUtils.addVolume( new VolumeInfo([project,]outVolume, outPartition, "outLabel"), new JobConf());
     ```

     In the preceding commands, project and label are optional, and the current project and default label are used by default. If multiple input and output files are used, labels are used to distinguish the files from each other. Authorization is required before you access the volume files of other projects.

   - Configure parameters to specify the volume and partition of the input and output files. If multiple input or output files are used, separate the parameters with commas (,).

```
set odps.sql.volume.input[/output].desc = [<project>.]<table>.<partition>[:<label>];
```

2. Call the following method by using a context object in the map and reduce steps to write data to the distributed file system or write data stream input and output files:

```
context.getOutputVolumeFileSystem();
```

1.6.19. Analysis of the mapping between SQL input and output fields

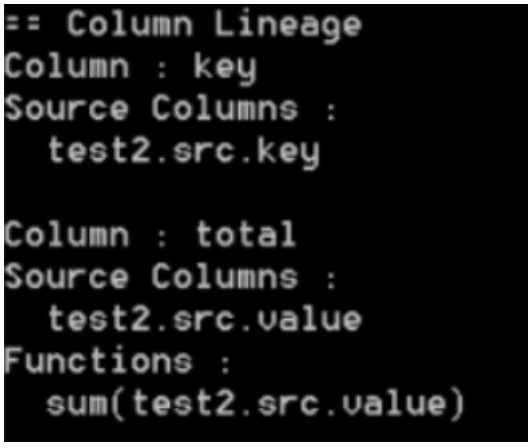
1.6.19.1. Features

MaxCompute SQL provides the feature of analyzing the mapping between SQL input and output fields.

This feature is to calculate the fields in the input and output tables based on field mapping. Example:

```
select key, sum(value) as total from  src group by key;
```

The following result is returned.



Two columns are returned: key and total. The key column corresponds to the src.key column of the input table. The total column corresponds to the src.value column of the input table.

1.6.19.2. Usage notes

This topic describes how to use the feature of analyzing the mapping between input and output fields.

Output format

Field mapping analysis supports human-readable and JSON formats. You can use the `set odps.sql.select.output.format=HumanReadable/json` flag to specify the output format.

SDK-based field mapping analysis

Examples

```
Odps odps = initOdps();
// To perform analysis, use LineageTask.
LineageTask task = new LineageTask("task_name", "select * from dual;");
// Optional. Use the preceding flag to specify the output format.
Map<String, String> settings = new LinkedHashMap<>();
settings.putIfAbsent("odps.sql.select.output.format", "json");
task.setProperty("settings", JSON.toJSONString(settings));
// Submit code to the server for field mapping analysis.
Instance instance = odps.instances().create(task);
System.out.println(instance.getId());
String logView = odps.logview().generateLogView(instance, 72);
System.out.println(logView);
instance.waitForSuccess();
// Obtain the analysis result.
System.out.println(instance.getTaskResults().get("task_name"));
```

odpscmd-based field mapping analysis

Examples

CLI mode: Use the -X parameter for field mapping analysis.

```
./bin/odpscmd.bat -X D:\lineage.q
```

Interactive mode: After you enter the interactive mode of odpscmd, you can use the preceding flag to specify the output format. The usage method is similar to that used to commit SQL jobs.

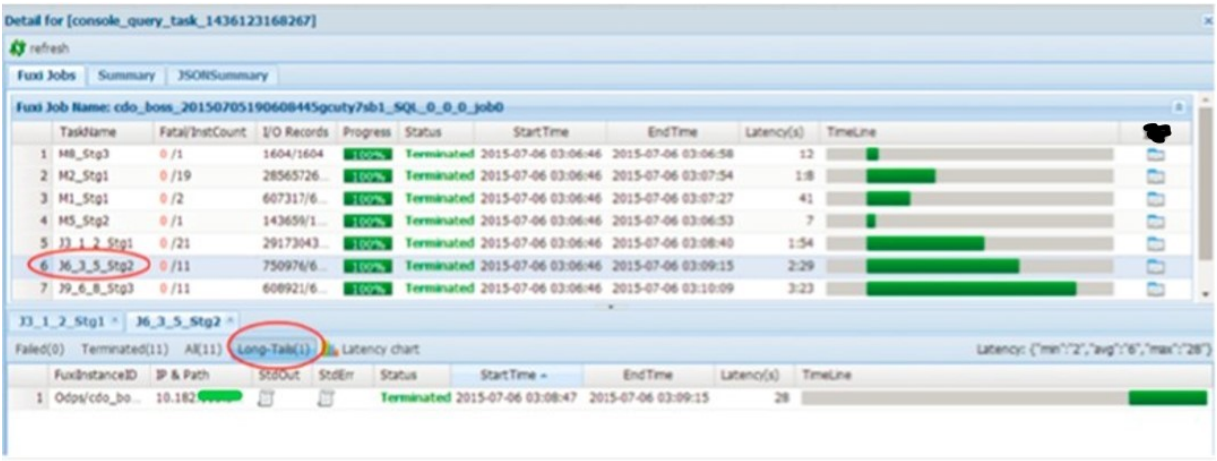
```
odps@ lineage_test>set odps.sql.task.mode=LINEAGE;
OK
odps@ lineage_test>set odps.sql.select.output.format=Humanreadable;
OK
odps@ lineage_test>select * from dual;
== Column Lineage
Column : id
Source Columns :
 test2.dual.id
```

1.6.20. Common MaxCompute SQL errors and solutions

1.6.20.1. Data skew

1.6.20.1.1. Overview

For a running job instance where the min, max, and avg values for the parameters time, input records, and output records are imbalanced (for example, max is much greater than avg), a data skew problem may have occurred. You can check the log view to locate the data skew problem, as shown in the following figure.



The Long Tails tab of each task shows the instance where the data skew occurred. The root cause of data skew is that the amounts of data processed by some instances are much higher than that processed by other instances, causing the running time of these instances to exceed the average time of other instances. As a result, the entire job slows down.

You can reduce the data skew of different SQL data types using different methods.

1.6.20.1.2. GROUP BY skew

Possible cause: The unbalanced distribution of GROUP BY keys causes data skew in the Reduce step.

Solution: Enable the group skew prevention parameter before running SQL statements:

```
set odps.sql.groupby.skewindata=true
```

> **Note** If this parameter is set to true, the system adds random factors to the shuffle hash algorithm and adds a new task to prevent data skew.

1.6.20.1.3. DISTRIBUTE BY skew

Possible cause: Using constants for full-table sorting in DISTRIBUTE BY mode will result in data skew at the Reduce end.

Solution: Avoid the preceding operation.

1.6.20.1.4. JOIN skew

Possible cause: The unbalanced distribution of join on keys (such as a large number of repeated keys in multiple JOIN tables) causes surging Cartesian product data in some JOIN instances, which results in data skew.

Solution: The solutions to different scenarios are as follows:

- If there are small tables on both sides of 'join', perform 'map join' instead of 'join'.
- The skewed key can be dealt with by using individual logic. For example, a large amount of NULL data in keys on both sides of a table results in skew. In this case, you need to filter out the NULL data before performing the JOIN operation or replacing NULL values with random values by using the CASE WHEN clause, and then do JOIN operation.
- If you do not want to change SQL statements, set the following parameters to enable automatic optimization on MaxCompute:

```
set odps.sql.skewinfo=tab1:(col1, col2)[(v1, v2), (v3, v4), ...]
set odps.sql.skewjoin=true;
```

1.6.20.1.5. MULTI-DISTINCT skew

Possible cause: Multiple DISTINCT keywords aggravate the GROUP BY skew problem.

Solution: You can use a two-layer GROUP BY to smooth the skew.

1.6.20.1.6. Data skew caused by misuse of dynamic partitioning

Possible cause: If dynamic partitioning is enabled, and there are K map instances and N target partitions, a number of small files (K * N) may be generated. A large amount of small files can greatly increase the management workload of the file system. Therefore, the following configuration takes effect by default:

```
set odps.sql.reshuffle.dynamicpt=true;
```

It introduces an additional level of ReduceTask to allow one or more reduce instances to write data to the same target partition. This prevents too many small files from being generated. However, dynamic partition shuffle may cause data skew.

Solution: If there are only a few target partitions, the system will not generate many small files. In this case, you can run the following command to disable the preceding function, or disable dynamic partitioning:

```
set odps.sql.reshuffle.dynamicpt=false;
```

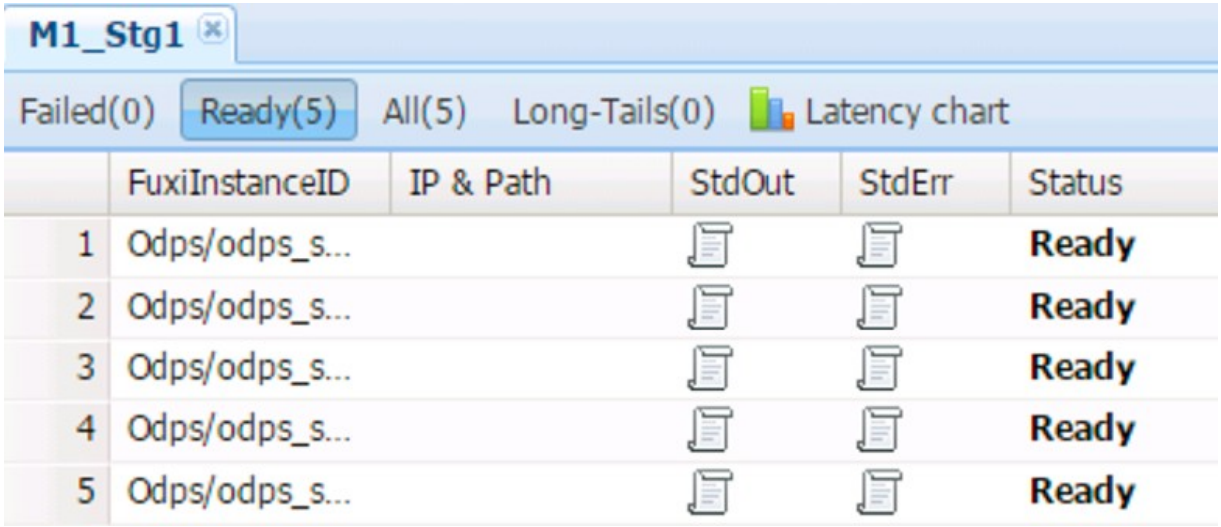
1.6.20.2. Quota and resource usage

Computing resources in MaxCompute may be insufficient sometimes because of improper planning and use of cluster resources.

In general, tasks lacking computing resources have two characteristics, one of which is that the task gets stuck with the output remained at a certain stage. For example, in the following figure, the progress of the M1_Stg1 task has stayed at 0% (because R2_1_Stg1 depends on M1_Stg1, it stays at 0% until M1_Stg1 ends).

```
2016-01-29 13:52:09 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:14 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:19 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:24 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:29 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:34 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:39 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:44 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:49 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:54 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:52:59 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:04 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:09 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:15 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:20 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:25 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:30 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:35 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:40 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:45 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:50 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
2016-01-29 13:53:55 M1_Stg1_job0:0/0/5[0%]      R2_1_Stg1_job0:0/0/1[0%]
```

The other characteristic is that the task remains in "Ready" state in the Logview (as shown in the following figure) (a "Ready" task is awaiting allocation of resources; a "Waiting" task is waiting for completion of the dependent task). The "Ready" state indicates that the resources for running these stand-by task instances are insufficient. Once the instances obtain the necessary resources, they resume operating and change to "Running" state.



Each task is split into subtasks based on the execution plan and shown in a DAG, and each subtask invokes multiple instances to execute the computation concurrently. In general, the resources required for invoking an instance are a 1-core CPU and 2 GB of memory. A quota group is assigned to each project for reasonable resource allocation. The quota group determines the maximum amount of resources (CPU and memory) that can be used by all jobs in the project concurrently. Once the resource usage for simultaneously running tasks reaches the limit of the quota group, the tasks are stuck due to insufficient resources.

There are two methods to solve this problem:

- Run the tasks in idle periods.
- Increase the quota group for the project (handled by OAM personnel).

1.6.20.3. MaxCompute storage optimization tips

Partition tables reasonably

MaxCompute supports the concept of partitioning in a table. A partition refers to the specified partition space in the creation of a table; that is, a few fields in the specified table as the partition columns. In most cases, you can consider a partition as a directory in a file system. MaxCompute divides each value of the partition column into a partition (directory). Users can specify multi-level partitions (use multiple fields of the table as partition columns). Multi-level partitions are like multi-level directories. If you specify the name of the partition that you want to access when using the data, then only the corresponding partition are read, avoiding a full table scan. This improves the processing efficiency and reduces costs.

Example of a partitioning statement: `create table src (key string, value bigint) partitioned by (pt string);`. In this example, `select * from src where pt='20160901';` specifies the partitioning format. MaxCompute takes only the data in the "20160901" partition as the input when generating a query plan.

Example of a non-partitioning statement: `select * from src where key = 'MaxCompute';` scans the entire table.

Partitioning is usually based on date or geographical region. You may also set partitions based on your business requirements. Example:

```
create table if not exists sale_detail(
shop_name string,
customer_id string,
total_price double)
partitioned by (sale_date string,region string);
-- Create a two-level partitioned table, in which sale_date is level-1 partition, and region level-2 partiti
on.
```

Set table lifecycle reasonably

Storage space on MaxCompute is precious. You can set the life cycle of a table according to data usage. MaxCompute will delete expired data to save storage space.

Example: Run the `create table test3 (key boolean) partitioned by (pt string, ds string) lifecycle 100;` command to create a table with a lifecycle of 100. If the latest modification time of this table or partition was more than 100 days ago, the table or partition will be deleted.

> **Notice** The lifecycle takes a partition as the smallest unit, so for a partitioned table, if some partitions reach the lifecycle threshold, they will be deleted directly. Partitions that have not reached the lifecycle threshold are not be affected.

Run the `alter table table_name set lifecycle days;` command to modify the lifecycle of an existing table.

Archive cold data

Some data need to be preserved either permanently or for a long period of time, but the frequency of access decreases over time. When the use frequency is very low, you can archive the data. The archive function saves data with RAID. Data is not simply stored as three copies. By using the Cauchy Reed-Solomon algorithm, data is stored as six copies of the original data plus three parity blocks. This improves the effective storage ratio from 1:3 to 1:1.5. In addition, MaxCompute uses the bzip2 algorithm to archive tables with a higher compression ratio than other algorithms. Combining the two algorithms reduces storage usage by more than 70%.

Archiving command format is as below:

```
ALTER TABLE table_name [PARTITION(partition_name='partition_value')] ARCHIVE;
```

Example:

```
alter table my_log partition(ds='20140101') archive;
```

Merge small files

In the reduce calculation or real-time tunnel data collection, a large number of small files are generated. Too many small files may cause the following problems:

- Many instances are occupied because a single instance can process only a small number of files. This results in a waste of resources, affecting the overall execution performance.
- The file system becomes larger, while the use ratio of disk space becomes smaller.

Currently, there are two alternative ways to merge small files: ALTER merge mode and SQL merge mode:

- The ALTER merge mode merges files through 'console' command. The command format is as follows:

  ```
  ALTER TABLE tablename [PARTITION] MERGE SMALLFILES;
  ```

- Set control parameters after SQL execution is complete. Run `odps.task.merge.enabled=true;` to determine whether it is necessary to merge small files. If so, start FuxiJob to merge these files.

1.6.20.4. UDF OOM error

Some jobs will report the OOM error during running. The error message is as follows:

```
FAILED: ODPS-0123144: Fuxi job failed - WorkerRestart errCode:9,errMsg:SigKill(OOM), usually caused by OOM(out of memory)
```

This problem can be solved by setting the UDF runtime parameters:

```
odps.sql.mapper.memory=3072;
set odps.sql.udf.jvm.memory=2048;
set odps.sql.udf.python.memory=1536;
```

1.6.21. Appendix

1.6.21.1. Escape character

String constants in MaxCompute SQL can be enclosed in single or double quotation marks, in double quotation marks enclosed in single quotation marks, or in single quotation marks enclosed in double quotation marks. Otherwise, they must be expressed with an escape character. Examples of correct expressions: "I'm a happy coder!" and 'I\'m a happy coder!'.

In MaxCompute SQL, the backslash (\) is an escape character, which expresses the special character in a string or interprets the character that follows as the character itself. When a string constant is read, if the backslash is followed by three valid octal digits in the range from 001 to 177, the system converts the ASCII values into the corresponding characters. The following table lists the mappings between escape sequences and represented characters.

Escape sequences

Escape sequence	Represented character
\b	Backspace
\t	Tab
\n	Newline
\r	Carriage return
\'	Single quote
\"	Double quote
\ \	Backslash
\;	Semicolon
\Z	Control-Z
\0 or \00	Terminator

Example:

```
select length('a\tb') from dual;
-- The result is 3, indicating that the string contains three characters, with "\t" regarded as one charac
ter. Any character following the escape sequence is interpreted as the character itself.
```

```
select 'a\ab',length('a\ab') from dual;
-- The result is 'aab', with a length of 3. "\a" is interpreted as an ordinary "a".
```

1.6.21.2. LIKE matching

In LIKE matching, "%" indicates matching any number of characters; "_" indicates matching a single character. If the character "%" or "_" needs to be matched, escape conversion is required. "\%" indicates matching "%", and "_" indicates matching "_".

> **Note** For the character set of strings, MaxCompute SQL currently supports the UTF-8 character set. Data that is encoded in a different format may result in incorrect calculations.

1.6.21.3. Regular expressions

MaxCompute SQL adopts the PCRE library for regular expressions. Matching is performed character by character. The supported metacharacters are as follows:

- ^: the beginning of a row
- $: the end of a row
- .: any character
- *: matches zero or multiple times.
- +: matches once or multiple times.
- ?: matches a modifier. If this character follows any one of other delimiters (*, +, ?, {n}, {n,}, or {n,m}), the match is lazy. In the lazy mode, as few strings as possible are matched. In the default greedy mode, as many searched strings as possible are matched zero times or once.
- A|B: A or B
- (abc)*: matches the abc sequence zero or multiple times.
- {n} or {m,n}: the number of matches
- [ab]: matches any character in the brackets.
- [^ab]: ^ represents NOT. This metacharacter matches any character that is neither a nor b.
- \: the escape sequence
- \n: n represents digit 1 to 9. This metacharacter specifies backward reference.
- \d: digit
- \D: non-digit
- [::]: POSIX character set
  - [[:alnum:]]: letter or digit in the range of [a-zA-Z0-9]
  - [[:alpha:]]: letter in the range of [a-zA-Z]
  - [[:ascii:]]: ASCII character in the range of [\x00-\x7F]
  - [[:blank:]]: space and tab in the range of [\t]

- [[:cntrl:]]: control character in the range of [\x00-\x1F\x7F]
- [[:digit:]]: digit in the range of [0-9]
- [[:graph:]]: any character except space in the range of [\x21-\x7E]
- [[:space:]]: space in the range of [\t\r\n\v\f]
- [[:print:]]: [:graph:] and [:space:] in the range of [\x20-\x7E]
- [[:lower:]]: lowercase letter in the range of [a-z]
- [[:punct:]]: punctuation in the range of [][!"#$%&()*+,./:;<=>? @\^_`{|}~-]
- [[:upper:]]: uppercase letter in the range of [A-Z]
- [[:xdigit:]]: hexadecimal character in the range of [A-Fa-f0-9]

The system uses a backslash (\) as the escape character, so a backslash (\) in a regular expression indicates second escape. For example, the string to be matched by the regular expression is "a+b". The plus sign (+) is a special character in regex, and must be escaped to obtain the string "a+b". However, the system needs to escape the first backslash (escape character) before it can be read by regex. Hence, the expression to match "a+b" is "a\\+b".

The following example assumes that there is a table named test_dual:

```
select 'a+b' rlike 'a\\+b' from test_dual;
+------+
_c1 |
+------+
true |
+------+
```

In extreme cases, to match the character "\", which is a special character in the regular engine, the expression must be "\\". The system must perform an escape on the expression, so it is expressed as "\\\".

```
select 'a\\b', 'a\\b' rlike 'a\\\\b' from test_dual;
+-----+------+
_c0 | _c1 |
+-----+------+
a\b | true |
+-----+------+
```

Note

- If a MaxCompute SQL statement contains "a\b", 'a\b' is displayed in the output because MaxCompute escapes the expression.
- If a string contains a tab or tab character, the system reads '\t' and stores it as one character. Therefore, it is a common character in the regular expression mode.

```
select 'a\tb', 'a\tb' rlike 'a\tb' from test_dual;
+---------+------+
_c0 | _c1 |
+---------+------+
a b | true |
+---------+------+
```

1.6.21.4. Reserved words

The following are all reserved words in MaxCompute SQL. Do not use these words to name tables, columns, or partitions. Otherwise, an error is returned. Reserved words are case-insensitive.

```
% & && ( ) * +. / ; < <= <> = > >= ? ADD AFTER ALL ALTER ANALYZE AND ARCHIVE ARRAY AS ASC BEFORE
BETWEEN BIGINT BINARY BLOB BOOLEAN BOTH BUCKET BUCKETS BY CASCADE CASE CAST CFILE CHANGE
CLUSTER CLUSTERED CLUSTERSTATUS COLLECTION COLUMN COLUMNS COMMENT COMPUTE CONCATENA
TE CONTINUE CREATE CROSS CURRENT CURSOR DATA DATABASE DATABASES DATE DATETIME DBPROPER
TIES DEFERRED DELETE DELIMITED DESC DESCRIBE DIRECTORY DISABLE DISTINCT DISTRIBUTE DOUBLE DR
OP ELSE ENABLE END ESCAPED EXCLUSIVE EXISTS EXPLAIN EXPORT EXTENDED EXTERNAL FALSE FETCH FI
ELDS FILEFORMAT FIRST FLOAT FOLLOWING FORMAT FORMATTED FROM FULL FUNCTION FUNCTIONS GRA
NT GROUP HAVING HOLD_DDLTIME IDXPROPERTIES IF IMPORT IN INDEX INDEXES INPATH INPUTDRIVER INP
UTFORMAT INSERT INT INTERSECT INTO IS ITEMS JOIN KEYS LATERAL LEFT LIFECYCLE LIKE LIMIT LINES LO
AD LOCAL LOCATION LOCK LOCKS LONG MAP MAPJOIN MATERIALIZED MINUS MSCK NOT NO_DROP NULL
OF OFFLINE ON OPTION OR ORDER OUT OUTER OUTPUTDRIVER OUTPUTFORMAT OVER OVERWRITE PARTI
TION PARTITIONED PARTITIONPROPERTIES PARTITIONS PERCENT PLUS PRECEDING PRESERVE PROCEDURE
PURGE RANGE RCFILE READ READONLY READS REBUILD RECORDREADER RECORDWRITER REDUCE REGEXP
RENAME REPAIR REPLACE RESTRICT REVOKE RIGHT RLIKE ROW ROWS SCHEMA SCHEMAS SELECT SEMI SEQ
UENCEFILE SERDE SERDEPROPERTIES SET SHARED SHOW SHOW_DATABASE SMALLINT SORT SORTED SSL
STATISTICS STORED STREAMTABLE STRING STRUCT TABLE TABLES TABLESAMPLE TBLPROPERTIES TEMP
ORARY TERMINATED TEXTFILE THEN TIMESTAMP TINYINT TO TOUCH TRANSFORM TRIGGER TRUE UNARCHI
VE UNBOUNDED UNDO UNION UNIONTYPE UNIQUEJOIN UNLOCK UNSIGNED UPDATE USE USING UTC UTC_T
MESTAMP VIEW WHEN WHERE WHILE
```

1.6.21.5. New data type settings

If you want to read a table that includes new data types, you are not required to add the `set setodps.sql.type.system.odps2=true;` flag. However, you must take note of the following points:

- If the flag is not added, the read data is implicitly converted into the original data type for all computations.
- If the flag is not added for integer constants, the BIGINT type is used, and an error message is returned.
- If you write data to a table and the data is in passthrough mode, you can choose not to add the new data type flag. However, if you want to calculate the data, an error is returned because the implicit data type conversion is invalid.

1.7. MaxCompute Tunnel

1.7.1. Overview

MaxCompute provides two types of channels for data uploads and downloads:

- **DataHub**: This channel is used to upload or download data in real time. It includes the OGG, Flume, Logstash, and Fluentd plug-ins.
- **Tunnel**: This channel is used to upload or download large amounts of data at a time. It includes the MaxCompute client, DataWorks, DTS, Sqoop, Kettle plug-in, and MaxCompute Migration Assist (MMA).

DataHub and Tunnel provide their own SDKs. The data upload and download tools derived from these SDKs meet the requirements of the most common scenarios in which data is migrated to the cloud. The tools also enable you to upload or download data in a variety of other scenarios.

Limits

- Limits on Tunnel-based data uploads:
  - You cannot run Tunnel commands to upload or download data of the ARRAY, MAP, or STRUCT types.
  - No limits are specified for the upload speed. The upload speed depends on the network bandwidth and server performance.
  - The number of retries is limited. If the number of retries exceeds the limit, the next block is uploaded. After data is uploaded, you can execute the select count(*) from table_name statement to check whether any data is lost.
  - By default, a project supports a maximum of 2,000 concurrent Tunnel connections.
  - On the server, the lifecycle of a session is 24 hours. A session can be shared among processes and threads on the server, but you must make sure that each block ID is unique.
  - MaxCompute ensures the validity of concurrent writes based on atomicity, consistency, isolation, durability (ACID).
- Limits on DataHub-based data uploads:
  - The size of each field cannot exceed its upper limit.

    > **Note** The size of a string cannot exceed 8 MB.

  - During an upload, multiple data records are packaged.

- Limits on TableTunnel SDK interfaces:
  - A block ID must be greater than or equal to 0 but less than 20,000. The size of the data that you want to upload in a block cannot exceed 100 GB.
  - The lifecycle of a session is 24 hours. If you want to transfer large amounts of data, more than 24 hours are required. In this case, we recommend that you transfer the data in multiple sessions.
  - The lifecycle of an HTTP request that corresponds to a RecordWriter is 120 seconds. If no data flows over an HTTP connection within 120 seconds, the server closes the connection.

1.7.2. Tunnel service connections

DataHub and Tunnel use different endpoints in different network environments. You must also select different endpoints when connecting to the service.

1.7.3. Selection of cloud data migration tools

MaxCompute provides a variety of data upload and download tools, which can be used in different cloud data migration scenarios. This topic describes the selection of data transmission tools in three typical scenarios.

Hadoop data migration

You can use Sqoop and DataWorks to migrate Hadoop data.

- When you use DataWorks, DataX is required.
- When you use Sqoop, a MapReduce job is executed on the original Hadoop cluster for distributed data transmission to MaxCompute.

Synchronization of data in a database

To synchronize data from a database to MaxCompute, you must select a tool based on the database type and synchronization policy.

- Use DataWorks for offline batch synchronization. DataWorks supports a wide range of database types, including MySQL, SQL Server, and PostgreSQL.
- Use the OGG plug-in for real-time synchronization of data in an Oracle database.
- Use DTS for real-time synchronization of data in an ApsaraDB for RDS database.

Log collection

You can use tools such as Flume, Fluentd, and Logstash to collect logs.

1.7.4. Introduction to the tools

MaxCompute supports a wide range of data upload and download tools. The source code for most of the tools can be found and maintained on the open-source community GitHub. You can select the appropriate tools to upload and download data based on the application scenario.

Alibaba Cloud DTplus products

- Data Integration of DataWorks (Tunnel)

Data Integration of DataWorks is a stable, efficient, and scalable data synchronization platform provided by Alibaba Cloud. It is designed to provide full offline and incremental real-time data synchronization, integration, and exchange services for the heterogeneous data storage systems on Alibaba Cloud.

Data synchronization tasks support the following data source types: MaxCompute, ApsaraDB for RDS (MySQL, SQL Server, and PostgreSQL), Oracle, FTP, AnalyticDB (ADS), OSS, ApsaraDB for Memcache, and DRDS.

- MaxCompute client (Tunnel)

  Based on the batch data tunnel SDK, the client provides built-in Tunnel commands for data upload and download.

- DTS (Tunnel)

  Data Transmission (DTS) is an Alibaba Cloud data service that supports data exchange between multiple data sources, such as Relational Database Management System (RDBMS), NoSQL, and Online Analytical Processing (OLAP) databases. It provides data transmission features, such as data migration, real-time data subscription, and real-time data synchronization.

  DTS supports data synchronization from ApsaraDB for RDS and MySQL instances to MaxCompute tables. Other data source types are not supported.

Open-source products

The projects corresponding to each product are open-sourced. You can visit aliyun-maxcompute-data-collectors to view details.

- Sqoop (Tunnel)

  Sqoop 1.4.6 on the community is further developed to provide enhanced MaxCompute support. It can import data from relational databases such as MySQL and data from HDFS or Hive to MaxCompute tables. It can also export data from MaxCompute tables to relational databases such as MySQL.

- Kettle (Tunnel)

  Kettle is an open-source ETL tool that is developed in Java. It can run on Windows, Unix, or Linux. It provides graphic interfaces for you to define data transmission topology by using drag-and-drop components.

- Flume (DataHub)
  - Apache Flume is a distributed and reliable system. It collects large volumes of log data from different data sources and then aggregates and stores the data in a centralized data storage.
  - The DataHub Sink plug-in of Apache Flume allows you to upload log data to DataHub in real time and archive the data in MaxCompute tables.

- Fluentd (DataHub)
  - Fluentd is an open-source software product. It collects logs, such as application logs, system logs, and access logs, from various sources. It allows you to use plug-ins to filter log data and store the data in different data processors, including MySQL, Oracle, MongoDB, Hadoop, and Treasure Data.
  - The DataHub plug-in of Fluentd allows you to upload log data to DataHub in real time and archive the data in MaxCompute tables.

- Logstash (DataHub)
  - Logstash is an open-source log collection and processing framework. The logstash-output-datahub plug-in allows you to import data to DataHub. This tool can be easily configured to collect and transmit data. It can be used together with MaxCompute or StreamCompute to easily create an all-in-one streaming data solution from data collection to analysis.
  - The DataHub plug-in of Logstash allows you to upload log data to DataHub in real time and archive the data in MaxCompute tables.
- OGG (DataHub)

  The DataHub plug-in of OGG allows you to incrementally synchronize data in the Oracle database to DataHub in real time and archive the data in MaxCompute tables.

1.7.5. Tunnel SDK overview

1.7.5.1. Overview

Data upload and download tools provided by MaxCompute are compiled based on the Tunnel SDK. This topic describes the major APIs of the Tunnel SDK.

The usage of the SDK varies according to the version. For specific information, see SDK Java Doc.

Major APIs

API	Description
TableTunnel	An entry class of the MaxCompute Tunnel service.
TableTunnel.UploadSession	A session that uploads data to a MaxCompute table.
TableTunnel.DownloadSession	A session that downloads data from a MaxCompute table.
InstanceTunnel	An entry class of the MaxCompute Tunnel service.
InstanceTunnel.DownloadSession	A session that downloads data from a MaxCompute instance. This session applies only to SQL instances that start with the SELECT keyword and are used to query data.

Note The tunnel endpoint supports automatic routing based on the MaxCompute endpoint settings.

1.7.5.2. TableTunnel

This topic describes the TableTunnel API.

Definition

Definition:

```
public class TableTunnel {
public DownloadSession createDownloadSession(String projectName, String tableName);
public DownloadSession createDownloadSession(String projectName, String tableName, PartitionSpec
partitionSpe c);
public UploadSession createUploadSession(String projectName, String tableName);
public UploadSession createUploadSession(String projectName, String tableName, PartitionSpec partit
ionSpec);
public DownloadSession getDownloadSession(String projectName, String tableName, PartitionSpec pa
rtitionSpec, String id);
public DownloadSession getDownloadSession(String projectName, String tableName, String id);
public UploadSession getUploadSession(String projectName, String tableName, PartitionSpec partition
Spec, String id);
public UploadSession getUploadSession(String projectName, String tableName, String id); public void s
etEndpoint(String endpoint);
}
```

Description:

- **Lifecycle**: the duration from the creation of the TableTunnel instance to the end of the program.
- TableTunnel provides a method to create UploadSession and DownloadSession objects. TableTunnel.UploadSession is used to upload data, and TableTunnel.DownloadSession is used to download data.
- A session refers to the process of uploading or downloading a table or partition. A session consists of one or more HTTP requests to Tunnel RESTful APIs.
- Upload sessions of TableTunnel use the INSERT INTO semantics. Multiple upload sessions of the same table or partition does not affect each other, and the data uploaded in each session is stored in an independent directory.
- In an upload session, each RecordWriter is matched with an HTTP request and is identified by a unique block ID. The block ID is the name of the file corresponding to the RecordWriter.
- If you use the same block ID to enable a RecordWriter multiple times in the same session, the data uploaded by the RecordWriter that calls the close() function last will overwrite all previous data. This feature can be used to retransmit data of a block when data upload fails.

API implementation process

1. The RecordWriter.write() function uploads your data as files to a temporary directory.
2. The RecordWriter.close() function moves the files from the temporary directory to the Data directory.
3. The session.commit() function moves each file in the Data directory to the directory where the corresponding table is located and updates the table metadata. This way, data moved into a table by the current task will be visible to the other MaxCompute tasks such as SQL and MapReduce.

API limits

- The value of a block ID must be greater than or equal to 0 and less than 20000. The size of data to be uploaded in a block cannot exceed 100 GB.
- A session is uniquely identified by its session ID. The lifecycle of a session is 24 hours. If your session times out due to the transfer of large volumes of data, you must transfer your data in multiple sessions.
- The lifecycle of an HTTP request corresponding to a RecordWriter is 120 seconds. If no data flows over an HTTP connection within 120 seconds, the server closes the connection.

  > **Note** HTTP has an 8 KB buffer. When you call the RecordWriter.write() function, your data may be saved to the buffer and no inbound traffic flows over the corresponding HTTP connection. In this case, you can call the TunnelRecordWriter.flush() function to forcibly flush data from the buffer.

- When you use a RecordWriter to write logs to MaxCompute, the RecordWriter may time out due to unexpected traffic fluctuations. Therefore, we recommend that you:
  - Do not use a RecordWriter for each data record. Otherwise, a large number of small files are generated, because each RecordWriter corresponds to a file. This affects the performance of MaxCompute.
- Do not use a RecordWriter to write data until the size of cached code reaches 64 MB.
- The lifecycle of a RecordReader is 300 seconds.

1.7.5.3. InstanceTunnel

This topic describes the InstanceTunnel API.

Definition:

```
public class InstanceTunnel{
 public DownloadSession createDownloadSession(String projectName, String instanceID);
 public DownloadSession createDownloadSession(String projectName, String instanceID, boolean limit
Enabled);
 public DownloadSession getDownloadSession(String projectName, String id);
 }
```

Parameter description:

- **projectName**: the name of a project.
- **instanceID**: the ID of an instance.

Limits: Although InstanceTunnel provides an easy way to obtain instance execution results, it is subject to the following permission limits to ensure data security:

- If the number of records does not exceed 10,000, all users who have the read permission on the specified instance can use InstanceTunnel to download the data. This is also applicable to the scenario of calling a Restful API to query data.
- If the number of records exceeds 10,000, only users who have the permission to read all the source tables from which the specified instance queries data can use InstanceTunnel to download the data.

1.7.5.4. UploadSession

This topic describes the UploadSession interface.

Definition

```
public class UploadSession {
UploadSession(Configuration conf, String projectName, String tableName, String partitionSpec) throws
TunnelException;
UploadSession(Configuration conf, String projectName, String tableName, String partitionSpec, String
uploadId) throws TunnelException;
public void commit(Long[] blocks); public Long[] getBlockList();
public String getId();
public TableSchema getSchema();
public UploadSession.Status getStatus(); public Record newRecord();
public RecordWriter openRecordWriter(long blockId);
public RecordWriter openRecordWriter(long blockId, boolean compress);
}
```

The following section describes the UploadSession interface.

Description

Item	Description
Lifecycle	The lifecycle of an upload instance starts when the instance is created and ends when data is uploaded.
Create a data upload instance	Create a data upload instance by calling a constructor method or by using TableTunnel. • The synchronous request mode is used. • The server creates a session for the data upload instance and generates a unique upload ID. You can run the getId command to obtain the upload ID.
Upload data	• The asynchronous request mode is used. • Call the openRecordWriter method to generate a RecordWriter. The blockId parameter identifies the data to upload and the position of the data in the table. The value of the blockId parameter ranges from 0 to 20000. If the upload fails, you can upload the data again based on the block ID.
View the status of an upload session	• The synchronous request mode is used. • Call the getStatus method to obtain the status of an upload session. • Call the getBlockList method to obtain the block IDs of successful upload sessions. You must check the block IDs of all upload sessions to identify failed upload sessions. Then, upload the data of failed sessions again.

Item	Description
Complete a data upload	• The synchronous request mode is used. • Call the commit(Long[] blocks) method. The blocks parameter indicates the list of the block IDs of successful upload sessions. The server verifies the list. • The verification enhances data accuracy. If the provided list of block IDs is different from the list on the server, an error is reported.
Status	• UNKNOWN: the initial state of a session. • NORMAL: An upload session is created. • CLOSING: The server sets the upload session to the CLOSING state before it calls the COMPLETE method to complete the data upload. • CLOSED: The data upload is complete. The data is moved to the directory of the result table. • EXPIRED: The upload session times out. • CRITICAL: An error occurs.

Notice

- Each block ID in an upload session must be unique. If you use a block ID to open a RecordWriter, write data, and then call the CLOSE and COMMIT methods, you cannot use this block ID to open another RecordWriter.
- The maximum size of a block is 100 GB. Make sure that the volume of data written to each block is greater than 64 MB. Otherwise, computing performance is severely reduced.
- The lifecycle of a session is 24 hours.
- Before you call the openRecordWriter method to write data, we recommend that you prepare data. A network action is triggered every time an openRecordWriter writes 8 KB of data. If no network actions are triggered within 120 consecutive seconds, the server closes the connection and the openRecordWriter becomes unavailable. If this happens, you must open a new openRecordWriter.
- The overwrite mode is added in the COMMIT method. You can use the overwrite mode to submit data. If you use the overwrite mode, the data in this commit overwrites the existing data of a table or partition.

  Notice If multiple sessions are concurrently executed and the overwrite mode is used to submit data, undefined behavior is generated. This may cause data inaccuracy. If you use the overwrite mode to submit data, you must control concurrent commits.

1.7.5.5. DownloadSession

This topic describes the DownloadSession class.

API definition:

```
public class DownloadSession {
DownloadSession(Configuration conf, String projectName, String tableName, String partitionSpec) thr
ows TunnelException
DownloadSession(Configuration conf, String projectName, String tableName, String partitionSpec, Stri
ng downloadId) throws TunnelException
public String getId()
public long getRecordCount() public TableSchema getSchema()
public DownloadSession.Status getStatus()
public RecordReader openRecordReader(long start, long count)
public RecordReader openRecordReader(long start, long count, boolean compress)
}
```

DownloadSession API description.

DownloadSession API

Parameter	Description
Lifecycle	From the creation of the Download instance to the end of the download process.
Purpose	Creates a Download instance by calling a constructor method or using TableTunnel. • Request mode: Synchronous. • The server creates a session for this Download and generates a unique download ID to mark the Download. The console can get data with a get ID. The operation has a high overhead. The server creates indexes for the data files. If many data files exist, the operation takes a long time. Then the server returns the total number of records, and starts concurrent downloads according to the number of records.
Download data	• Request mode: Asynchronous. • Call openRecordReader to generate a RecordReader instance. The Start parameter marks the start position of record for this download. The value of Start is equivalent to or greater than 0. The Count parameter marks the number of records for this download. The value of Count is greater than 0.
View the download process	• Request mode: Synchronous. • Call getStatus to get the download status.
Status	• UNKNOWN: the initial value that is set when the server creates a session. • NORMAL: The download object is successfully created. • CLOSED: The download session is completed. • EXPIRED: The download session times out.

1.7.5.6. TunnelBufferedWriter

This topic describes the TunnelBufferedWriter interface.

The upload process is complex due to limits on block management and connection timeout on the server. The Tunnel SDK provides an enhanced RecordWriter, TunnelBufferWriter, to simplify the upload process.

The TunnelBufferedWriter interface is defined as follows:

```
public class TunnelBufferedWriter implements RecordWriter {
   public TunnelBufferedWriter(TableTunnel.UploadSession session, CompressOption option) throws I
OException;
   public long getTotalBytes();
   public void setBufferSize(long bufferSize);
   public void setRetryStrategy(RetryStrategy strategy);
   public void write(Record r) throws IOException;
   public void close() throws IOException;
}
```

A TunnelBufferedWriter object is described as follows:

- **Lifecycle**: the duration from the time RecordWriter is created to the time the data upload ends.
- **TunnelBufferedWriter instance**: You can call the openBufferedWriter interface of UploadSession to create a TunnelBufferedWriter instance
- **Data upload**: When you call the Write interface, data is first written to the local cache. After the cache is full, the data is submitted to the server in batches to avoid connection timeout. In addition, if the upload fails, the system automatically retries the upload operation.
- **End upload**: Call the Close interface and then call the Commit interface of UploadSession to end the upload process.
- **Buffer control**: You can use the setBufferSize interface to modify the memory occupied by the buffer (in bytes), preferably 64 MB or more to prevent the server from generating too many small files, which may affect performance. The valid range is 1 MB to 1000 MB. The default value is 64 MB, which is recommended in most cases.
- **Retry policy settings**: You have three retry avoidance policies to choose from: EXPONENTIAL_BACKOFF, LINEAR_BACKOFF, and CONSTANT_BACKOFF. For example, the following code segment sets the Write retry count to 6. To avoid unnecessary retries, each retry is performed only after exponentially ascending intervals of 4s, 8s, 16s, 32s, 64s, and 128s by default.

  ```
  RetryStrategy retry
    = new RetryStrategy(6, 4, RetryStrategy.BackoffStrategy.EXPONENTIAL_BACKOFF)
  writer = (TunnelBufferedWriter) uploadSession.openBufferedWriter();
  writer.setRetryStrategy(retry);
  ```

> **Note** We recommend that you do not adjust the preceding settings.

1.7.6. Tunnel SDK example

1.7.6.1. Simple upload example

This topic provides a simple upload example of Tunnel SDK.

Example:

```
import java.io.IOException;
import java.util.Date;
import com.aliyun.odps.Column;
import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordWriter;
import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TunnelException;
import com.aliyun.odps.tunnel.TableTunnel.UploadSession;
public class UploadSample {
  private static String accessId = "<your access id>";
  private static String accessKey = "<your access Key>";
  private static String tunnelUrl = "<your tunnel endpoint>";
  private static String odpsUrl = "<your odps endpoint>";
  private static String project = "<your project>";
  private static String table = "<your table name>";
  private static String partition = "<your partition spec>";
  public static void main(String args[]) {
    Account account = new AliyunAccount(accessId, accessKey);
    Odps odps = new Odps(account);
    odps.setEndpoint(odpsUrl);
    odps.setDefaultProject(project);
    try {
      TableTunnel tunnel = new TableTunnel(odps);
      tunnel.setEndpoint(tunnelUrl);
      PartitionSpec partitionSpec = new PartitionSpec(partition);
      UploadSession uploadSession = tunnel.createUploadSession(project,
          table, partitionSpec);
```

```
    System.out.println("Session Status is : "
        + uploadSession.getStatus().toString());
    TableSchema schema = uploadSession.getSchema();
    // After data is prepared, run the Writer command to start writing data. The prepared data is writt
en to a block.
    // Writing a small volume of data to each block can result in a large number of small files.This great
ly reduces computing performance. We strongly recommend that you write at least 64 MB (and up to 10
0 GB) of data to each block.
    // You can estimate the total data volume based on the average data volume and record count. For
example, 64 MB < Average data volume x Record count < 100 GB.
    RecordWriter recordWriter = uploadSession.openRecordWriter(0);
    Record record = uploadSession.newRecord();
    for (int i = 0; i < schema.getColumns().size(); i++) {
      Column column = schema.getColumn(i);
      switch (column.getType()) {
        case BIGINT:
          record.setBigint(i, 1L);
          break;
        case BOOLEAN:
          record.setBoolean(i, true);
          break;
        case DATETIME:
          record.setDatetime(i, new Date());
          break;
        case DOUBLE:
          record.setDouble(i, 0.0);
          break;
        case STRING:
          record.setString(i, "sample");
          break;
        default:
          throw new RuntimeException("Unknown column type: "
              + column.getType());
      }
    }
    for (int i = 0; i < 10; i++) {
      // Write data to the server. A network transmission process is triggered each time 8 KB of data is
written.
      // If no data is transmitted for 120 seconds, the connection times out. The Writer command becom
es unavailable, and you must write data again.
      recordWriter.write(record);
```

```
recordWriter.write(record);
    }
    recordWriter.close();
    uploadSession.commit(new Long[]{0L});
    System.out.println("upload success!");
  } catch (TunnelException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
 }
}
```

1.7.6.2. Simple download example

This topic provides an example for the simple download function of Tunnel SDK.

Example:

```
import java.io.IOException; import java.util.Date;
import com.aliyun.odps.Column; import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec; import com.aliyun.odps.TableSchema; import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount; import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordReader; import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TableTunnel.DownloadSession; import com.aliyun.odps.tunnel.TunnelException;
public class DownloadSample {
private static String accessId = "<your access id>"; private static String accessKey = "<your access Key>";
private static String tunnelUrl = "<your tunnel endpoint>";
private static String odpsUrl = "<your odps endpoint>";
private static String project = "<your project>"; private static String table = "<your table name>";
private static String partition = "<your partition spec>";
public static void main(String args[]) {
Account account = new AliyunAccount(accessId, accessKey); Odps odps = new Odps(account); odps.setEndpoint(odpsUrl);
odps.setDefaultProject(project);
TableTunnel tunnel = new TableTunnel(odps); tunnel.setEndpoint(tunnelUrl);
PartitionSpec partitionSpec = new PartitionSpec(partition); try {
DownloadSession downloadSession = tunnel.createDownloadSession(project, table, partitionSpec);
System.out.println("Session Status is : "
```

```
+ downloadSession.getStatus().toString());
long count = downloadSession.getRecordCount(); System.out.println("RecordCount is: " + count);
RecordReader recordReader = downloadSession.openRecordReader(0, count);
Record record;
while ((record = recordReader.read()) ! = null) { consumeRecord(record, downloadSession.getSchema(
));
}
recordReader.close();
} catch (TunnelException e) { e.printStackTrace();
} catch (IOException e1) { e1.printStackTrace();
}
}
private static void consumeRecord(Record record, TableSchema schema) { for (int i = 0; i < schema.get
Columns().size(); i++) {
Column column = schema.getColumn(i); String colValue = null;
switch (column.getType()) { case BIGINT: {
Long v = record.getBigint(i);
colValue = v == null ? null : v.toString(); break;
}
case BOOLEAN: {
Boolean v = record.getBoolean(i); colValue = v == null ? null : v.toString(); break;
}
case DATETIME: {
Date v = record.getDatetime(i); colValue = v == null ? null : v.toString(); break;
}
case DOUBLE: {
Double v = record.getDouble(i); colValue = v == null ? null : v.toString(); break;
}
case STRING: {
String v = record.getString(i);
colValue = v == null ? null : v.toString(); break;
}
default:
throw new RuntimeException("Unknown column type: "
+ column.getType());
}
System.out.print(colValue == null ? "null" : colValue); if (i ! = schema.getColumns().size())
System.out.print("\t");
}
System.out.println();
}
```

```
}
```

1.7.6.3. Multithread upload example

This topic provides a multithread upload example of Tunnel SDK.

Example:

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import com.aliyun.odps.Column;
import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordWriter;
import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TunnelException;
import com.aliyun.odps.tunnel.TableTunnel.UploadSession;
class UploadThread implements Callable<Boolean> {
 private long id;
 private RecordWriter recordWriter;
 private Record record;
 private TableSchema tableSchema;
 public UploadThread(long id, RecordWriter recordWriter, Record record, TableSchema tableSchema) {
  this.id = id;
  this.recordWriter = recordWriter;
  this.record = record;
  this.tableSchema = tableSchema;
 }
 @Override
 public Boolean call() {
  for (int i = 0; i < tableSchema.getColumns().size(); i++) {
   Column column = tableSchema.getColumn(i);
   switch (column.getType()) {
    case BIGINT:
```

```
        record.setBigint(i, 1L);
        break;
      case BOOLEAN:
        record.setBoolean(i, true);
        break;
      case DATETIME:
        record.setDatetime(i, new Date());
        break;
      case DOUBLE:
        record.setDouble(i, 0.0);
        break;
      case STRING:
        record.setString(i, "sample");
        break;
      default:
        throw new RuntimeException("Unknown column type: "
            + column.getType());
    }
   }
   try {
    for (int i = 0; i < 10; i++) {
    // Write data to the server. A network transmission process is triggered each time 8 KB of data is w
ritten.
      // If no data is transmitted for 120 seconds, the connection times out. The Writer command becom
es unavailable and you must write data again.
      recordWriter.write(record);
    }
    recordWriter.close();
   } catch (IOException e) {
    e.printStackTrace();
    return false;
   }
   return true;
 }
}
public class UploadThreadSample {
 private static String accessId = "<your access id>";
 private static String accessKey = "<your access Key>";
 private static String tunnelUrl = "<your tunnel endpoint>";
 private static String odpsUrl = "<your odps endpoint>";
```

```
  private static String project = "<your project>";
  private static String table = "<your table name>";
  private static String partition = "<your partition spec>";
  private static int threadNum = 10;:
  public static void main(String args[]) {
    Account account = new AliyunAccount(accessId, accessKey);
    Odps odps = new Odps(account);
    odps.setEndpoint(odpsUrl);
    odps.setDefaultProject(project);
    try {
      TableTunnel tunnel = new TableTunnel(odps);
      tunnel.setEndpoint(tunnelUrl);
      PartitionSpec partitionSpec = new PartitionSpec(partition);
      UploadSession uploadSession = tunnel.createUploadSession(project,
          table, partitionSpec);
      System.out.println("Session Status is : "
          + uploadSession.getStatus().toString());
      ExecutorService pool = Executors.newFixedThreadPool(threadNum);
      ArrayList<Callable<Boolean>> callers = new ArrayList<Callable<Boolean>>();
      // After the data is prepared, open a writer to start multithread data writing.
      // Writing a small volume of data to each block can result in a large number of small files. This grea
tly affects computing performance. We recommend that you write at least 64 MB (and up to 100 GB) of
data to each block.
      // You can estimate the total data volume based on the average data volume and record count. For
example, 64 MB < Average data volume x Record count < 100 GB.
      for (int i = 0; i < threadNum; i++) {
        RecordWriter recordWriter = uploadSession.openRecordWriter(i);
        Record record = uploadSession.newRecord();
        callers.add(new UploadThread(i, recordWriter, record, uploadSession.getSchema()));
      }
      pool.invokeAll(callers);
      pool.shutdown();
      Long[] blockList = new Long[threadNum];
      for (int i = 0; i < threadNum; i++)
        blockList[i] = Long.valueOf(i);
      uploadSession.commit(blockList);
      System.out.println("upload success!");
    } catch (TunnelException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
```

```
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

1.7.6.4. Multithread download example

This topic provides a multithread download example of Tunnel SDK.

Example:

```
import java.io.IOException;
import java.util.ArrayList; import java.util.Date; import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException; import java.util.concurrent.ExecutorService; import jav
a.util.concurrent.Executors;
import java.util.concurrent.Future;
import com.aliyun.odps.Column; import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec; import com.aliyun.odps.TableSchema; import com.aliyun.odps.ac
count.Account;
import com.aliyun.odps.account.AliyunAccount; import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.RecordReader; import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TableTunnel.DownloadSession; import com.aliyun.odps.tunnel.TunnelEx
ception;
class DownloadThread implements Callable<Long> { private long id;
private RecordReader recordReader; private TableSchema tableSchema;
public DownloadThread(int id,
RecordReader recordReader, TableSchema tableSchema) { this.id = id;
this.recordReader = recordReader; this.tableSchema = tableSchema;
}
@Override
public Long call() {
Long recordNum = 0L; try {
Record record;
while ((record = recordReader.read()) ! = null) { recordNum++;
System.out.print("Thread " + id + "\t"); consumeRecord(record, tableSchema);
}
recordReader.close();
} catch (IOException e) { e.printStackTrace();
}
return recordNum;
```

```
}
private static void consumeRecord(Record record, TableSchema schema) { for (int i = 0; i < schema.get
Columns().size(); i++) {
Column column = schema.getColumn(i); String colValue = null;
switch (column.getType()) { case BIGINT: {
Long v = record.getBigint(i);
colValue = v == null ? null : v.toString(); break;
}
case BOOLEAN: {
Boolean v = record.getBoolean(i); colValue = v == null ? null : v.toString(); break;
}
case DATETIME: {
Date v = record.getDatetime(i); colValue = v == null ? null : v.toString(); break;
}
case DOUBLE: {
Double v = record.getDouble(i); colValue = v == null ? null : v.toString(); break;
}
case STRING: {
String v = record.getString(i);
colValue = v == null ? null : v.toString(); break;
}
default:
throw new RuntimeException("Unknown column type: "
+ column.getType());
}
System.out.print(colValue == null ? "null" : colValue); if (i ! = schema.getColumns().size())
System.out.print("\t");
}
System.out.println();
}
}
public class DownloadThreadSample {
private static String accessId = "<your access id>"; private static String accessKey = "<your access Key
>";
private static String tunnelUrl = "<your tunnel endpoint>";
private static String odpsUrl = "<your odps endpoint>";
private static String project = "<your project>"; private static String table = "<your table name>";
private static String partition = "<your partition spec>";
private static int threadNum = 10; public static void main(String args[]) {
Account account = new AliyunAccount(accessId, accessKey);
```

```
Odps odps = new Odps(account); odps.setEndpoint(odpsUrl); odps.setDefaultProject(project);
TableTunnel tunnel = new TableTunnel(odps); tunnel.setEndpoint(tunnelUrl);
PartitionSpec partitionSpec = new PartitionSpec(partition); DownloadSession downloadSession;
try {
downloadSession = tunnel.createDownloadSession(project, table, partitionSpec);
System.out.println("Session Status is : "
+ downloadSession.getStatus().toString());
long count = downloadSession.getRecordCount(); System.out.println("RecordCount is: " + count);
ExecutorService pool = Executors.newFixedThreadPool(threadNum); ArrayList<Callable<Long>> caller
s = new ArrayList<Callable<Long>>();
long start = 0;
long step = count / threadNum;
for (int i = 0; i < threadNum - 1; i++) {
RecordReader recordReader = downloadSession.openRecordReader( step * i, step);
callers.add(new DownloadThread( i, recordReader, downloadSession.getSchema()));
}
RecordReader recordReader = downloadSession.openRecordReader(step * (threadNum - 1), count
+ ((threadNum - 1) * step));
callers.add(new DownloadThread( threadNum - 1, recordReader, downloadSession.getSchema()));
Long downloadNum = 0L;
List<Future<Long>> recordNum = pool.invokeAll(callers); for (Future<Long> num : recordNum)
downloadNum += num.get(); System.out.println("Record Count is: " + downloadNum); pool.shutdown()
;
} catch (TunnelException e) { e.printStackTrace();
} catch (IOException e) { e.printStackTrace();
} catch (InterruptedException e) { e.printStackTrace();
} catch (ExecutionException e) { e.printStackTrace();
}
}
}
```

1.7.6.5. Example of uploading data by using BufferedWriter

This topic provides an example on how to upload data by using BufferedWriter of the Tunnel SDK.

Example:

```
//Initialize the code of MaxCompute and Tunnel.
RecordWriter writer = null;
TableTunnel.UploadSession uploadSession = tunnel.createUploadSession(projectName, tableName);
try {
  int i = 0;
  //Generate a TunnelBufferedWriter instance.
  writer = uploadSession.openBufferedWriter();
  Record product = uploadSession.newRecord();
  for (String item : items) {
    product.setString("name", item);
    product.setBigint("id", i);
    //Call the Write interface to write data.
    writer.write(product);
    i += 1;
  }
} finally {
  if (writer ! = null) {
    //Disable TunnelBufferedWriter.
    writer.close();
  }
}
//Commit the upload session to end the upload.
uploadSession.commit();
```

1.7.6.6. Example of uploading data by using BufferedWriter in multi-threaded mode

This topic provides an example on how to upload data by using BufferedWriter of the Tunnel SDK in multi-threaded mode.

Example:

```
class UploadThread extends Thread {
  private UploadSession session;
  private static int RECORD_COUNT = 1200;
  public UploadThread(UploadSession session) {
    this.session = session;
  }
  @Override
  public void run() {
    RecordWriter writer = up.openBufferedWriter();
    Record r = up.newRecord();
    for (int i = 0; i < RECORD_COUNT; i++) {
      r.setBigint(0, i);
      writer.write(r);
    }
    writer.close();
  }
};
public class Example {
  public static void main(String args[]) {
    //Initialize the code of MaxCompute and Tunnel.
    TableTunnel.UploadSession uploadSession = tunnel.createUploadSession(projectName, tableName);
    UploadThread t1 = new UploadThread(up);
    UploadThread t2 = new UploadThread(up);
    t1.start();
    t2.start();
    t1.join();
    t2.join();
    uploadSession.commit();
}
```

1.7.6.7. Examples of uploading and downloading complex data

This topic provides examples of uploading and downloading complex data by using the Tunnel SDK.

Upload complex data

Example:

```
RecordWriter recordWriter = uploadSession.openRecordWriter(0);
ArrayRecord record = (ArrayRecord) uploadSession.newRecord();
//Prepare data.
List arrayData = Arrays.asList(1, 2, 3);
Map<String, Long> mapData = new HashMap<String, Long>();
mapData.put("a", 1L);
mapData.put("c", 2L);
List<Object> structData = new ArrayList<Object>();
structData.add("Lily");
structData.add(18);
//Import data to a record.
record.setArray(0, arrayData);
record.setMap(1, mapData);
record.setStruct(2, new SimpleStruct((StructTypeInfo) schema.getColumn(2).getTypeInfo(), structData
));
//Write the record.
recordWriter.write(record);
```

Download complex data

Example:

```
RecordReader recordReader = downloadSession.openRecordReader(0, 1);
//Read a record.
ArrayRecord record1 = (ArrayRecord)recordReader.read();
//Obtain data of the ARRAY type.
List field0 = record1.getArray(0);
List<Long> longField0 = record1.getArray(Long.class, 0);
//Obtain data of the MAP type.
Map field1 = record1.getMap(1);
Map<String, Long> typedField1 = record1.getMap(String.class, Long.class, 1);
//Obtain data of the STRUCT type.
Struct field2 = record1.getStruct(2);
```

Example of upload and download

Example:

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
```

```
import java.util.List;
import java.util.Map;
import com.aliyun.odps.Odps;
import com.aliyun.odps.PartitionSpec;
import com.aliyun.odps.TableSchema;
import com.aliyun.odps.account.Account;
import com.aliyun.odps.account.AliyunAccount;
import com.aliyun.odps.data.ArrayRecord;
import com.aliyun.odps.data.RecordReader;
import com.aliyun.odps.data.RecordWriter;
import com.aliyun.odps.data.SimpleStruct;
import com.aliyun.odps.data.Struct;
import com.aliyun.odps.tunnel.TableTunnel;
import com.aliyun.odps.tunnel.TableTunnel.UploadSession;
import com.aliyun.odps.tunnel.TableTunnel.DownloadSession;
import com.aliyun.odps.tunnel.TunnelException;
import com.aliyun.odps.type.StructTypeInfo;
public class TunnelComplexTypeSample {
  private static String accessId = "<your access id>";
  private static String accessKey = "<your access Key>";
  private static String odpsUrl = "<your odps endpoint>";
  private static String project = "<your project>";
  private static String table = "<your table name>";
  //Partitions in a partitioned table, such as "pt=\'1\',ds=\'2\'".
  //If the table is not a partitioned table, you do not need to execute the following statement.
  private static String partition = "<your partition spec>";
  public static void main(String args[]) {
    Account account = new AliyunAccount(accessId, accessKey);
    Odps odps = new Odps(account);
    odps.setEndpoint(odpsUrl);
    odps.setDefaultProject(project);
    try {
      TableTunnel tunnel = new TableTunnel(odps);
      PartitionSpec partitionSpec = new PartitionSpec(partition);
      //---------- Upload data ----------
      //Create an upload session for the table.
      //The table schema is {"col0": ARRAY<BIGINT>, "col1": MAP<STRING, BIGINT>, "col2": STRUCT<name:
STRING,age:BIGINT>}.
      UploadSession uploadSession = tunnel.createUploadSession(project, table, partitionSpec);
      //Obtain the table schema.
      TableSchema schema = uploadSession.getSchema();
```

```
    //Enable the RecordWriter.
    RecordWriter recordWriter = uploadSession.openRecordWriter(0);
    ArrayRecord record = (ArrayRecord) uploadSession.newRecord();
    //Prepare data.
    List arrayData = Arrays.asList(1, 2, 3);
    Map<String, Long> mapData = new HashMap<String, Long>();
    mapData.put("a", 1L);
    mapData.put("c", 2L);
    List<Object> structData = new ArrayList<Object>();
    structData.add("Lily");
    structData.add(18);
    //Import data to a record.
    record.setArray(0, arrayData);
    record.setMap(1, mapData);
    record.setStruct(2, new SimpleStruct((StructTypeInfo) schema.getColumn(2).getTypeInfo(), structD
ata));
    //Write the record.
    recordWriter.write(record);
    //Disable the writer.
    recordWriter.close();
    //Commit the upload session to end the upload.
    uploadSession.commit(new Long[]{0L});
    System.out.println("upload success!") ;
    //---------- Download data ----------
    //Create a download session for the table.
    //The table schema is {"col0": ARRAY<BIGINT>, "col1": MAP<STRING, BIGINT>, "col2": STRUCT<name:
STRING, age:BIGINT>}.
    DownloadSession downloadSession = tunnel.createDownloadSession(project, table, partitionSpec)
;
    schema = downloadSession.getSchema();
    //Enable the record reader. In the example, one record is read.
    RecordReader recordReader = downloadSession.openRecordReader(0, 1);
    //Read a record.
    ArrayRecord record1 = (ArrayRecord)recordReader.read();
    //Obtain data of the ARRAY type.
    List field0 = record1.getArray(0);
    List<Long> longField0 = record1.getArray(Long.class, 0);
    //Obtain data of the MAP type.
    Map field1 = record1.getMap(1);
    Map<String, Long> typedField1 = record1.getMap(String.class, Long.class, 1);
    //Obtain data of the STRUCT type.
```

```
    // Obtain data of the STRUCT type.
    Struct field2 = record1.getStruct(2);
    System.out.println("download success!") ;
   } catch (TunnelException e) {
    e.printStackTrace();
   } catch (IOException e) {
    e.printStackTrace();
   }
  }
}
```

1.7.7. Appendix

1.7.7.1. Tunnel upload/download FAQ

This topic describes frequently asked questions (FAQs) about tunnel upload and download.

What is MaxCompute Tunnel?

Tunnel is data channel of MaxCompute, you are available to upload or download data through Tunnel to or from MaxCompute. You can upload and download only table data (excluding view data) with MaxCompute Tunnel.

Can block IDs be repeated?

Each block ID in an Upload session must be unique. After a block ID is used to start RecordWriter in an upload session, data is written, and the session is closed and committed, this block ID cannot be used to start another RecordWriter. A maximum of 20,000 blocks are supported, with the block IDs ranging from 0 to 19999.

Is there a limit on block size?

The maximum size of a block is 100 GB. We strongly recommend that 64 MB or more data is written into each block. Each block is a file. A file smaller than 64 MB is a small file. Excessive small files will affect the computing performance.

Can a session be shared? Does a session have a life cycle?

Each session has a 24-hour life cycle on the server. It can be used within 24 hours after being created, and can be shared among processes or threads. The block ID of each session must be unique. The procedure for distributed uploading is as follows: **Create a session > Evaluate data volume > Assign blocks (for example, thread 1 uses blocks 0-100 and thread 2 uses blocks 100-200) > Prepare data > Upload data > Commit all blocks with data written.**

How to process write/read timeout or I/O exceptions?

During data uploading, a network action is triggered every time the Writer writes 8 KB data. If no network action is triggered within 120 seconds, the server closes the connection and the Writer becomes unavailable. You have to start a new Writer.

The Reader in data downloading works in a similar way. If no network I/O occurs for a period of time, the connection is closed. We suggest that you run Read without inserting any interfaces from other systems.

Which languages of SDK are available for MaxCompute Tunnel?

MaxCompute Tunnel provides the Java and C++ editions of SDK.

Does MaxCompute Tunnel allow multiple consoles to upload the same table at the same time?

Yes.

Is MaxCompute Tunnel suitable for batch uploading or stream uploading?

MaxCompute Tunnel is more suitable for batch uploading.

Are partitions required for data uploading through MaxCompute Tunnel?

Yes, MaxCompute Tunnel does not automatically build partitions.

What is the relationship between Dship and MaxCompute Tunnel?

Dship is a tool that uploads and downloads data through MaxCompute Tunnel.

Does data uploaded with MaxCompute Tunnel append to the existing file or overwrite the data?

The uploaded data appends to the file.

What is the routing function of MaxCompute Tunnel?

The routing function allows the Tunnel SDK to get the tunnel endpoint by setting MaxCompute. That is, you can run the Tunnel SDK properly by setting the endpoint of MaxCompute.

What is the preferred size of a block when data is uploaded with MaxCompute Tunnel?

The block size depends on factors such as the network situation, real-time performance requirement, data usage, and number of small files in a cluster. If a large volume of data is uploaded continuously, the preferred block size is 64-256 MB. If the data is uploaded in a batch once every day, the block size can be 1 GB.

Why is the timeout error often reported during data downloading with MaxCompute Tunnel?

This may have occurred due to an endpoint error. Check the endpoint configuration. A simple method is to run telnet to check the network connectivity to the endpoint.

Why does the following error occur during data downloading with MaxCompute Tunnel?

```
You have NO privilege 'odps:Select' on {acs:odps:*:projects/XXX/tables/XXX}. project 'XXX' is protecte
d
```

The data protection function has been enabled for the project. Only the project owner has the right to transfer data from one project to another if the project data is protected.

Why does the following error occur during data uploading with MaxCompute Tunnel?

```
ErrorCode=FlowExceeded, ErrorMessage=Your flow quota is exceeded. **
```

The maximum number of concurrent requests is exceeded. By default, MaxCompute Tunnel allows a maximum of 2,000 concurrent upload and download requests (quota). Each request, once it is sent, occupies one quota unit until it ends. Try the following solutions:

- Put the system to sleep, and try again after it awakes.
- Change the concurrency quota to a greater number for the project after obtaining the forecast flow pressure from the administrator.
- Report the problem to the project owner to check and control the top requests occupying a large quota.

1.7.7.2. Common tunnel error codes

This topic describes common tunnel error codes.

Common tunnel error codes are as follows.

Common error codes

Error code	Error	Processing recommendations
NoSuchPartition	The partition does not exist.	Tunnel doesn't create partitions, you need to create partitions and then upload or download.
InvalidProjectTable	Invalid project name or table name.	Check related names.
NoSuchProject	The project does not exist.	Check related names.
NoSuchTable	The table does not exist.	Check related names.
StatusConflict	The session expires or has been committed.	Re-create the session and upload it.
MalformedDataStream	Data format error.	Normally created because network is disconnected, or Schema and Table are different.
InvalidPartitionSpec	Invalid partition information.	Check partition information. An example of a correct value is pt='1',ct='2017'.

Error code	Error	Processing recommendations
InvalidRowRange	Invalid designated row range, normally it exceeds the max. size or it is 0.	Check related parameters.
Unauthorized	Account information error.	Normally it is wrong AccessId or AccessKey, or local device time has a 15-minute gap with server.
DataStoreError	Storage error.	Contact the administrator.
NoPermission	No permission normally because no related permission or IP whitelist has been set.	Check whether permission is correct.
MissingPartitionSpec	Missing partition information, partition table operation must carry partition information.	Add partition information.
TableModified	Data in the table is modified by other tasks while upload or download.	Re-create the session and re-try.
FlowExceeded	Exceed concurrency quota limit.	Check and control volume of concurrency. If it is needed to add concurrency, please contact the project owner or administrator to evaluate flow pressure.
InvalidResourceSpec	Normally because the project, table or partition information is different from the session.	Check related information and re-try.
MethodNotAllowed	Method is not allowed, normally try to export the view.	Exporting the view is not supported currently.
InvalidColumnSpec	Invalid column information.	Normally it is column name error while download designated column.
DataVersionConflict	It is cross-cluster coping.	Re-try later.
InternalServerError	Internal error.	Re-try or contact the administrator.

1.8. MaxCompute MapReduce

1.8.1. Overview

1.8.1.1. MapReduce

MaxCompute provides a MapReduce programming API. You can use the API to write MapReduce programs to process MaxCompute data.

MapReduce is a distributed data processing model initially proposed by Google. It later gained extensive attention in the industry and was widely used in a variety of business scenarios.

A MapReduce program processes data in two stages: the Map stage and the Reduce stage. It executes the Map stage first and then the Reduce stage. Although you can define the processing logics of Map and Reduce, they need to follow the conventions of the MapReduce framework.

The following is a detailed procedure of how MapReduce processes data:

1. Before you formally start Map, ensure that partition is set for input data. The input data is divided into equal-sized blocks, which are partitions. Each partition is processed as the input of a single Map worker so that multiple Map workers can work together.
2. After partitioning, multiple Map Workers start working simultaneously. Each Map Worker reads its respective shard, computes the shard, and works out the result to Reduce.

   > **Note** During data output, each Map worker needs to specify one key for each output data. The key decides the Reduce worker for which the data is targeted. Multiple keys may correspond to a single Reduce worker. Data of the same key is sent to the same Reduce worker, and a single Reduce worker may receive data with different keys.

3. Before entering the Reduce stage, the MapReduce framework will sort the data Key values to make data with the same Key values adjacent. If you specify Combiner, the framework will call Combiner and aggregate data with the same Key.

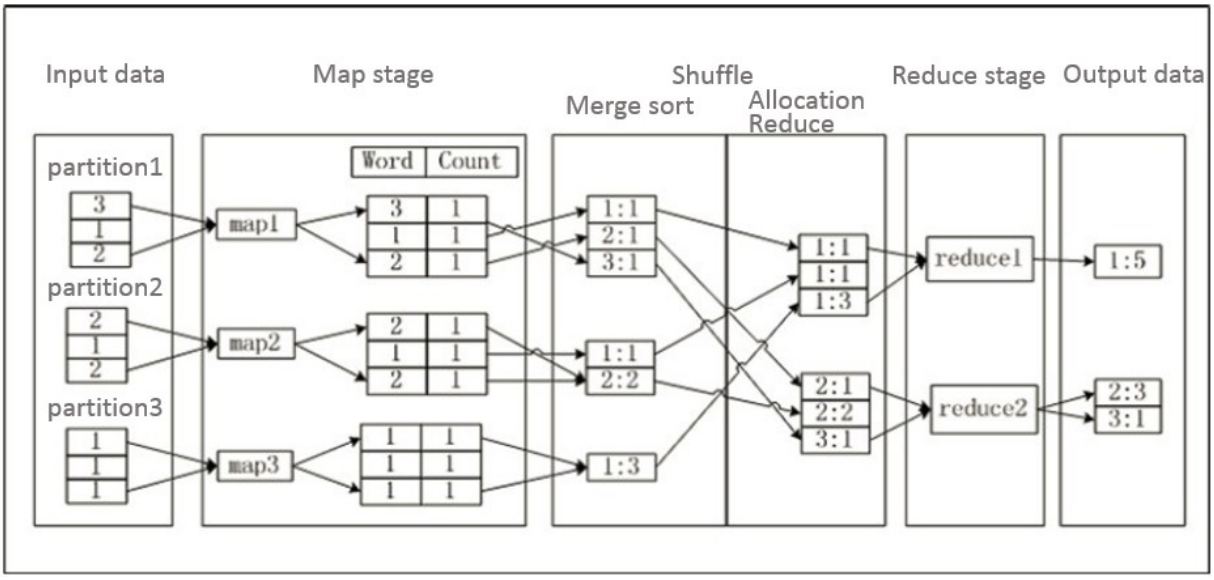
   > **Note** You can customize the Combiner logic. Unlike the typical MapReduce framework protocol, Unlike the typical MapReduce framework protocol, MaxCompute requires the input and output parameters be consistent with those of Reduce. This process is generally called Shuffle.

4. When entering the Reduce stage, data with same Key will be in the same Reduce Worker. A single Reduce Worker may receive data from multiple Map Workers. Each Reduce worker performs the Reduce operation on multiple values with the same key. Finally, the multiple data entries with 1 Key will become 1 Value after the Reduce operation.

> **Note** The preceding section is only a brief introduction to MapReduce. For more details, see related documentation.

The following uses WordCount as an example to explain the related concepts of MaxCompute MapReduce in different stages.

Assume there is a file a.txt, and in each line in the text is a digit. You need to count the number of times each digit appears. Each number is called a Word, and the number of its occurrences is called the Count. To use MaxCompute MapReduce for this purpose, perform the steps shown in the following figure.



1. Partition a.txt data and use data in each partition as input of a single Map worker.
2. For the Map processing input, the Count parameter is set to 1 for each obtained number. The Word|Count pair is output as a Word data key.
3. At the start of the Shuffle stage, the output of each Map worker is sorted by key value (Word value). MapReduce performs the COMBINER operation on the sorted outputs, combining data of the same key (Word value) to form a new Word|Count pair. The process is called combine sorting.
4. Later in the Shuffle stage, data is sent to the Reduce side. The Reduce Worker sorts the received data again depending on Key value.
5. During the Reduce stage, each Reduce Worker uses the same logic as Combiner while processing data, and adds the Count with the same Key value (Word value) to obtain the output result.

> **Note** Because all the MaxCompute data is saved in the table, the input and output of MaxCompute MapReduce can only be a table. Customizing the output format is not permitted, and similar file system APIs are not provided.

1.8.1.2. Extended MapReduce

In a traditional MapReduce model, data must be stored in a distributed file system (such as an HDFS or a MaxCompute table) after each round of MapReduce operations. A typical MapReduce application is composed of multiple MapReduce jobs. Data is written to a disk after each job is completed. Subsequent map tasks usually only read data once to prepare for the following Shuffle stage. This results in redundant I/O operations.

The computing scheduling logic of MaxCompute supports more complicated programming models. In the preceding case, a reduce operation can be followed by the next reduce operation without having a map operation in between. An extended MapReduce model is provided. This model supports any number of reduce operations after map, such as Map-Reduce-Reduce.

Hadoop Chain Mapper and Chain Reducer also support similar serialized map or reduce operations. However, they are essentially different from the extended MapReduce (MR2) model. Chain Mapper and Chain Reducer are based on the traditional MapReduce model. They support one or more mapper operations, not reducer operations, after the original mapper or reduce operation. One benefit is that you can reuse the preceding mapper business logic by splitting a map or reduce operation into multiple mapper stages. This, however, does not change the underlying scheduling or I/O model.

1.8.1.3. Open-source compatibility with MapReduce

MaxCompute offers a set of native MapReduce programming models and interfaces. The inputs and outputs for these interfaces are MaxCompute tables, and the data is organized to be processed in the record format.

However, MaxCompute MapReduce APIs differ from Hadoop MapReduce APIs. Traditionally, to migrate your Hadoop MapReduce jobs to MaxCompute, you need to rewrite the MapReduce code, compile and debug the code by using MaxCompute APIs, compress the final code into a JAR package, and upload the package to the MaxCompute platform. This process is tedious and labor-intensive for development and testing. It was expected that the original Hadoop MapReduce code can be used on the MaxCompute platform with little or no modification at all.

To achieve this purpose, the MaxCompute platform provides a plug-in to adapt Hadoop MapReduce to MaxCompute MapReduce. With this plug-in, Hadoop MapReduce jobs are compatible with MaxCompute at the binary level to a certain extent. You can specify some configurations without modifying the code and then run the original Hadoop MapReduce JAR packages on MaxCompute. Click here to download the plug-in. This plug-in is in the testing stage and does not support custom comparators or key types.

In the following section, a WordCount program is used as an example to introduce the basic usage of this plug-in.

> **Note**
>
> - For more information about open-source compatibility, see Compatibility with Hadoop MapReduce.
> - For more information about the Hadoop MapReduce SDK, see the MapReduce official documentation.
> - The code in the example is for reference only. You need to modify it based on your business needs.

Download the plug-in

Click here to download the plug-in. The package name is openmr_hadoop2openmr-1.0.jar.

> **Note** This JAR package contains the dependencies with Hadoop 2.7.2. To avoid version conflicts, you must not include Hadoop dependencies in the JAR packages of your jobs.

Prepare a JAR package

Compile and export the WordCount JAR package (wordcount_test.jar). The source code of the WordCount program is as follows:

```
package com.aliyun.odps.mapred.example.hadoop;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
import java.util.StringTokenizer;
public class WordCount {
  public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
      Context context
    ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
```

```
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

Prepare test data

1. Create input and output tables.

   ```
   create table if not exists wc_in(line string);
   create table if not exists wc_out(key string, cnt bigint);
   ```

2. Use Tunnel to import data to the input table.

   The following content in the data.txt file needs to be imported:

   ```
   hello maxcompute
   hello mapreduce
   ```

3. You can run the following command on the MaxCompute client to import data from data.txt to wc_in:

   ```
   tunnel upload data.txt wc_in;
   ```

Configure the mapping between HDFS file paths and MaxCompute tables

The configuration file is named wordcount-table-res.conf.

```
{
 "file:/foo": {
  "resolver": {
   "resolver": "com.aliyun.odps.mapred.hadoop2openmr.resolver.TextFileResolver",
   "properties": {
      "text.resolver.columns.combine.enable": "true",
      "text.resolver.seperator": "\t"
   }
  },
  "tableInfos": [
   {
    "tblName": "wc_in",
    "partSpec": {},
    "label": "__default__"
   }
  ],
  "matchMode": "exact"
 },
 "file:/bar": {
  "resolver": {
   "resolver": "com.aliyun.odps.mapred.hadoop2openmr.resolver.BinaryFileResolver",
   "properties": {
      "binary.resolver.input.key.class" : "org.apache.hadoop.io.Text",
      "binary.resolver.input.value.class" : "org.apache.hadoop.io.LongWritable"
   }
  },
  "tableInfos": [
   {
    "tblName": "wc_out",
    "partSpec": {},
    "label": "__default__"
   }
  ],
  "matchMode": "fuzzy"
 }
}
```

The preceding configuration is a JSON file that describes the mapping between HDFS file paths and MaxCompute tables. You need to configure both the input and output. Each HDFS file path matches three configuration items: resolver, tableInfos, and matchMode. The configuration items are described as follows:

- **resolver**: specifies how to process data in files. Currently, the following two built-in resolvers are available: com.aliyun.odps.mapred.hadoop2openmr.resolver.TextFileResolver and com.aliyun.odps.mapred.hadoop2openmr.resolver.BinaryFileResolver. After you specify the resolver name, you must configure properties for the resolver to support data parsing.
  - **TextFileResolver**: regards the input or output as plaintext if the data is of plaintext type. When you configure an input resolver, you must configure the text.resolver.columns.combine.enable and text.resolver.seperator properties. When text.resolver.columns.combine.enable is set to true, all columns in the input table are combined into a single string based on the delimiter specified by text.resolver.seperator. Otherwise, the first two columns in the input table are used as the key and value fields.
  - **BinaryFileResolver**: converts binary data into a data type that is supported by MaxCompute, such as BIGINT, BOOLEAN, and DOUBLE. When you configure an output resolver, you must configure the binary.resolver.input.key.class and binary.resolver.input.value.class properties. binary.resolver.input.key.class defines the key type of the intermediate result, and binary.resolver.input.value.class defines the value type.
- **tableInfos**: specifies the MaxCompute table that corresponds to HDFS. At present, only the tblName parameter is configurable. The partSpec and label parameters must be set to the values the same as those in the preceding example.
- **matchMode**: specifies the path matching mode. It can be set to exact or fuzzy. You can use a regular expression in fuzzy mode to match the HDFS input path.

Submit a job

Use the MaxCompute command line tool odpscmd to submit the job. Run the following command in odpscmd:

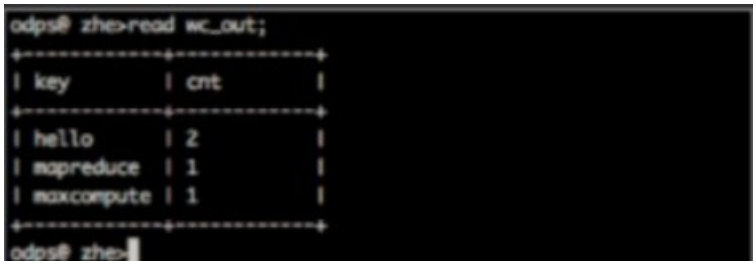
```
jar -DODPS_HADOOPMR_TABLE_RES_CONF=./wordcount-table-res.conf -classpath hadoop2openmr-1.0.j
ar,wordcount_test.jar com.aliyun.odps.mapred.example.hadoop.WordCount /foo/bar;
```

Note

- wordcount-table-res.conf: the mapped configuration file configured with /foo/bar.
- wordcount_test.jar: the JAR package of your Hadoop MapReduce program.
- com.aliyun.odps.mapred.example.hadoop.WordCount: the class name of the job that you want to run.
- /foo/bar: the path on HDFS, which is mapped to wc_in and wc_out in the JSON configuration file.
- After you configure the mapping, you must import the Hadoop HDFS input file to wc_in for MapReduce computing by using the data integration function of DataX or DataWorks, and export the result table wc_out to your HDFS output directory */bar*.
- Before you run the preceding command, make sure that hadoop2openmr-1.0.jar, wordcount_test.jar, and wordcount-table-res.conf have been stored in the current directory of odpscmd. Otherwise, you must make the relevant changes when you specify the configuration and -classpath.

The following figure shows the running process.

After the job running process is complete, you can view the result table wc_out to check whether the job is successful and whether the results meet expectations.



1.8.2. Features

1.8.2.1. Run command

The MaxCompute client provides a jar command for running MapReduce jobs.

Command syntax:

```
Usage: jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS]
-conf <configuration_file> Specify an application configuration file
-classpath <local_file_list> classpaths used to run mainClass
-D <name>=<value> Property value pair, which will be used to run mainClass
-local Run job in local mode
-resources <resource_name_list> file/table resources used in mapper or reducer, separate by comma
```

The following table describes the parameters.

Parameters

Parameter	Description
-conf <configuration file>	Indicates a JobConf file.
-classpath <local_file_list>	Indicates the classpath for local execution. It specifies the local paths (including relative path and absolute path) of the jar packet where the main function is located.
-D <prop_name>=<prop_value>	Indicates the java attribute of <mainClass> in local execution. You can define multiple attributes.
-local	Indicates that the MapReduce job is run locally. It is mainly used for program debugging.
-resources <resource_name_list>	Declares the resources used by a MapReduce job. Typically, you must specify the name of the resource where the Map or Reduce function is located in resource_name_list. **Notice** If the Map or Reduce function reads from other MaxCompute resources, you must add the names of these resource to resource_name_list. Multiple resources are separated by commas. If you want to use resources in another project, you must append PROJECT_NAME/resources/ before the resource name, such as -resources otherproject/resources/resfile.

Note The preceding optional parameters are included in <GENERIC_OPTIONS>.

You can use the -conf option to specify the JobConf file. This file can affect the settings of JobConf in the SDK. For more information about JobConf, see the introduction of MapReduce core interfaces. The following is an example of a JobConf file.

Example:

```
<configuration>
<property>
<name>import.filename</name>
<value>resource.txt</value>
</property>
</configuration>
```

In the preceding example, the JobConf file is used to define a variable named import.filename. The value of this variable is resource.txt. You can use the JobConf API in MapReduce to obtain the value of this variable. You can also use the JobConf API in the SDK for the same purpose.

Example:

```
jar -resources mapreduce-examples.jar -classpath mapreduce-examples.jar
org.alidata.odps.mr.examples.WordCount wc_in wc_out
add file data/src.txt
jar -resources src.txt,mapreduce-examples.jar -classpath mapreduce-examples.jar org.alidata.odps.mr
.examples.WordCount wc_in wc_out
add file data/a.txt
add table wc_in as test_table add jar work.jar
jar -conf odps-mapred.xml -resources a.txt,test_table,work.jar
-classpath work.jar:otherlib.jar
-D import.filename=resource.txt org.alidata.odps.mr.examples.WordCount args
```

1.8.2.2. Concepts

1.8.2.2.1. MapReduce

Map and Reduce functions support setup and cleanup methods for their corresponding map() and reduce() methods. A setup method is called prior to the Map() or Reduce() method. Each worker calls it once. A cleanup method is called after the Map() or Reduce() method. Each worker calls it once.

> **Note** For more information about the usage example, see Example program.

1.8.2.2.2. Sorting

Columns in the key records output by map can be set as sort columns. Custom comparators are not supported. You can select a few sort columns as group columns. Custom group comparators are not supported. Sort columns are generally used to sort user data, while group columns are used for secondary sorting.

> **Note** For detailed examples, see Secondary sorting source code.

1.8.2.2.3. Partition

MaxCompute supports partition columns and custom partitioners. Partition columns take precedence over custom partitioners. Partitioners are used to allocate Map output data to different Reduce workers based on the partition logic.

1.8.2.2.4. Combiner

The Combiner function combines adjacent records in the Shuffle stage. You can choose to use the Combiner function based on your business logic. The Combiner function is an optimized MapReduce computing framework. The Combiner logic is the same as the Reduce logic. After map outputs data, the framework merges data with the same key value on the map side locally.

1.8.2.2.5. Submit a job

This topic describes how to use the MaxCompute client to run and submit a MapReduce job.

The MaxCompute client provides a JAR command for running MapReduce jobs. Syntax:

```
jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS];
    -conf <configuration_file>        Specify an application configuration file
    -resources <resource_name_list>    file\table resources used in mapper or reducer, seperate by co
mma
    -classpath <local_file_list>      classpaths used to run mainClass
    -D <name>=<value>                 Property value pair, which will be used to run mainClass
    -l                          Run job in local mode
```

Example:

```
jar -conf \home\admin\myconf -resources a.txt,example.jar -classpath ..\lib\example.jar:.\other_lib.jar
-D java.library.path=.\native;
```

<GENERIC_OPTIONS> includes the following options, which are all optional:

- **-conf <configuration file>**: specifies the JobConf file. This file can affect the settings of JobConf in the SDK.
- **-resources <resource_name_list>**: declares the resources used for running the MapReduce job. You must specify the names of the resources in which the Map or Reduce function is located in resource_name_list.

  > **Note** If the Map or Reduce function reads from other MaxCompute resources, you must add the names of these resources to resource_name_list. Separate multiple resources with commas (,). If you want to use resources in another project, you must append PROJECT/resources/ to the beginning of the resource name, such as -resources otherproject/resources/resfile.

- **-classpath <local_file_list>**: specifies the classpath for local execution. It is used to specify the local paths (including relative and absolute paths) of the JAR package where the main function is located. Package names are separated with default file delimiters of the system. In most cases, semicolons (;) are used in Windows, and commas (,) are used in Linux. If you run a MapReduce job on a cloud server, separate package names with commas (,).

  > **Note** The main function and Map/Reduce function are usually written in the same package. In this case, the -resources and -classpath options both contain information such as mapreduce-examples.jar. However, -resources references the Map or Reduce function and is executed in a distributed environment, while -classpath references the main function and is executed locally with the specified JAR package saved in a local directory.

- **-D <name>=<value>**: specifies the Java attribute of <mainClass> during local execution. You can define multiple attributes.
- **-l** specifies that the MapReduce job is run locally. It is mainly used for program debugging.

The following code is an example of the JobConf file:

```
<configuration>
    <property>
      <name>import.filename</name>
      <value>resource.txt</value>
    </property>
  </configuration>
```

In the preceding example, the JobConf file is used to define a variable named import.filename. The value of this variable is resource.txt. You can use the JobConf API in MapReduce to obtain the value of this variable. You can also use the JobConf API in the SDK for the same purpose.

Example:

```
add jar data\mapreduce-examples.jar;
  jar -resources mapreduce-examples.jar -classpath data\mapreduce-examples.jar
    org.alidata.odps.mr.examples.WordCount wc_in wc_out;
  add file data\src.txt;
  add jar data\mapreduce-examples.jar;
  jar -resources src.txt,mapreduce-examples.jar -classpath data\mapreduce-examples.jar
    org.alidata.odps.mr.examples.WordCount wc_in wc_out;
  add file data\a.txt;
  add table wc_in as test_table;
  add jar data\work.jar;
  jar -conf odps-mapred.xml -resources a.txt,test_table,work.jar
    -classpath data\work.jar:otherlib.jar
    -D import.filename=resource.txt org.alidata.odps.mr.examples.WordCount args;
```

1.8.2.2.6. Input and output

- MaxCompute MapReduce supports the following build-in data types: Bigint, Double, String, Datetime, Boolean, Decimal, Tinyint, Smallint, Int, Float, Varchar, Timestamp, Binary, Array, Map, and Struct. MapReduce does not support custom data types.
- MapReduce supports input from multiple tables with different schemas. You can use the map function to obtain the table information corresponding to the current record.
- MapReduce supports NULL as input, but does not support views as input.
- Reduce can write output to different tables or different partitions of a table. The target tables can have different schemas. Each output is identified by a label. An output is not labeled by default. At least one output is generated.

> Note For examples, see Example programs.

1.8.2.2.7. Read data from resources

You can use the Map or Reduce function to read data from MaxCompute resources. Any Map or Reduce worker loads resources to the memory for you to write code.

> Note For a detailed example, see Resource utilization example.

1.8.2.2.8. Run MapReduce tasks locally

You can specify the -local parameter in the jar command to enable local debugging by simulating the running of a local MapReduce instance. During local running, the client downloads from MaxCompute the metadata and data of the input table, required resources, and the metadata of the output table needed for local debugging. The client saves the data to a local directory named warehouse. When MapReduce running is complete, MapReduce saves the computing results to a file in the warehouse directory. If the input table and required resources are already downloaded to the local warehouse directory, MapReduce directly references the data and files in the warehouse directory the next time it runs, instead of downloading the data again.

A MapReduce instance that is running locally may start multiple map or reduce processes, which run serially rather than concurrently. The simulated running process is different from an actual distributed running process in the following aspects:

- Input table row count: Up to 100 rows of data can be downloaded.
- Resource usage: In a distributed environment, MaxCompute limits the size of referenced resources. For more information, see Application limits. During local running, the size of resources is unlimited.
- Security limits: MaxCompute MapReduce and UDF programs running in a distributed environment are subject to Java sandbox restrictions. There is no such restrictions during local running.

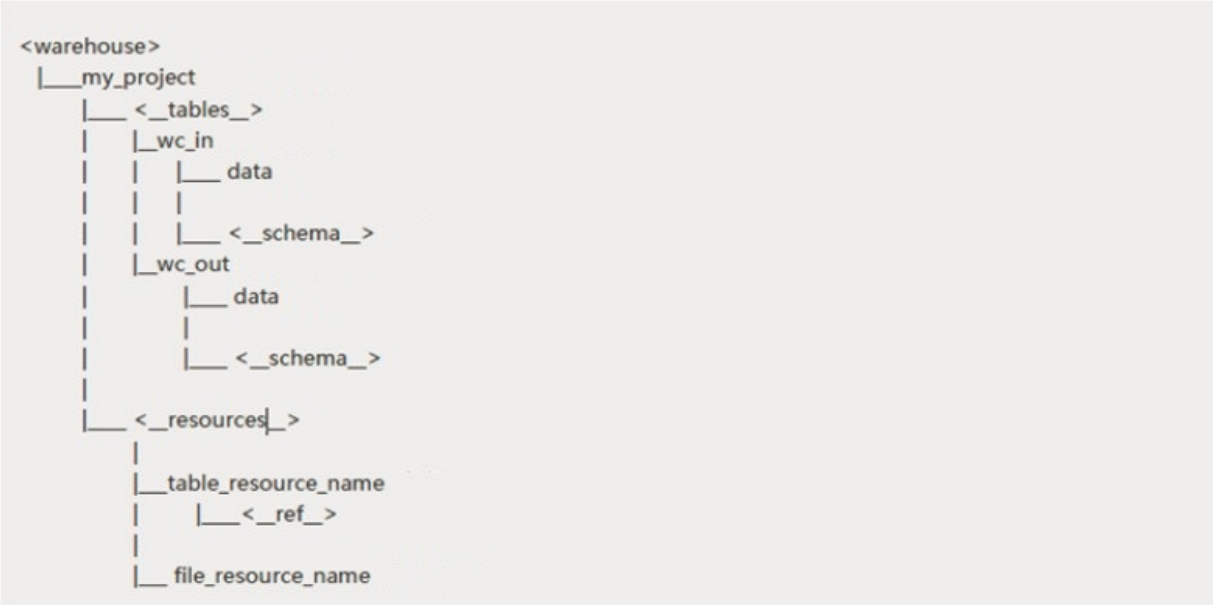
The following is a simple local running example.

Example:

```
odps@my_project> jar -l com.aliyun.odps.mapred.example.WordCount wc_in wc_out;
 Summary:
counters: 10
map-reduce framework combine_input_groups=2 combine_output_records=2 map_input_bytes=4 map_i
nput_records=1 map_output_records=2 map_output_[wc_out]_bytes=0 map_output_[wc_out]_records=0
reduce_input_groups=2 reduce_output_[wc_out]_bytes=8 reduce_output_[wc_out]_records=2
OK
```

> Note For a WordCount example, see WordCount example.

If you run the local debugging command for the first time, a directory named warehouse is created in the current path after the command is complete. The following figure shows the directory structure of warehouse.



Directories of the same level as my_project are projects. wc_in and wc_out are data tables. The table file data that you read or write using the jar command is downloaded to directories of this level. The schema file contains the metadata of a table in the following format:

```
project=local_project_name
table=local_table_name
columns=col1_name:col1_type,col2_name:col2_type
partitions=p1:STRING,p2:BIGINT
-- This field is not required in this example.
```

> **Note** Separate the name and type of a column with a colon (:), and separate columns with commas (,). You need to declare the project name and table name at the top of the schema files as projectname.tablename. Separate the declaration from column definition with a comma (,). The data file stores table data. The number of columns and data values must match the definition in the schema file. Separate columns with commas (,).

Example of wc_in schema file:

```
my_project.wc_in,key:STRING,value:STRING
```

Example of the corresponding data file:

```
0,2
```

> **Note** The client downloads the metadata and part of the data of a table from MaxCompute, and saves the data to the preceding files. The next time you run this example, the client directly uses the data in the wc_in directory, instead of downloading it again. Note that you can download data from MaxCompute only when running MapReduce locally. If you use the Eclipse plugin for local debugging, you cannot download data from MaxCompute.

Example of wc_out schema file:

```
my_project.wc_out,key:STRING,cnt:BIGINT
```

Example of the corresponding data file:

```
0,1
2,1
```

> **Note** The client downloads the metadata of we_out from MaxCompute, and saves the data to the schema file. The data file is generated to store the local running results. You can also compile schema and data files, and save them in the corresponding table directory. Then, when you run MapReduce locally, the client detects these files in the table directory, and will not download data from MaxCompute. The local table directory does not have to correspond to an actual table in MaxCompute.

The following table compares the features of Hadoop MapReduce with MaxCompute MapReduce.

Feature comparison

Feature	Hadoop MapReduce	MaxCompute MapReduce
Task progress report	The map stage calculates the read data volume. The Reduce step monitors the progress of various phases. Based on this information, the overall task progress can be estimated. The client can obtain real-time task progress. Task progress is not controlled by users.	Real-time task progress reporting is supported in a different way.
Statistics	Custom statistics, real-time summary, and real-time updates are supported. Real-time statistics updates are not controlled by users.	Real-time statistics updates are not supported during runtime. Calculation and summary are performed after a task is completed.
File compression	Users have an option to specify compressed storage.	File compression is not supported. Users cannot directly store files.
Speculative execution	/	Speculative execution is enabled by default, and cannot be configured by users.

Feature	Hadoop MapReduce	MaxCompute MapReduce
End of task notification	The server notifies the client of task completion.	No notifications are provided. When using the SDK to submit tasks, users must poll task status constantly until the tasks are completed.

1.8.3. SDK introduction

1.8.3.1. Major API overview

Major APIs

API	Description
MapperBase	User-defined Map functions must inherit this class. It converts the record objects in the input table into key-value pairs, and outputs them to the Reduce stage or directly to the result table by skipping the Reduce stage. The jobs that skip the Reduce stage and directly output calculation results are also called Map-Only jobs.
ReducerBase	User-defined Reduce functions must inherit this class. It reduces a set of values associated with a key.
TaskContext	One of the input parameters of multiple member functions of MapperBase and ReducerBase. It contains the contextual information of task execution.
JobClient	It is used to submit and manage jobs, including the blocking mode (synchronous) and non-blocking mode (asynchronous).
RunningJob	Job runtime objects for tracking running MapReduce job instances.
JobConf	Describes the configuration of a MapReduce task. The JobConf object is generally defined in the main program (main function). Then, JobClient submits the job to MaxCompute.

1.8.3.2. API description

1.8.3.2.1. MapperBase

The following table lists the major function APIs.

Major APIs

API	Description
void cleanup(TaskContext context)	The Map method is called after the map stage ends.
void map(long key, Record record, TaskContext context)	Map method for processing records in the input table.

API	Description
void setup(TaskContext context)	The Map method is called before the map stage begins.

1.8.3.2.2. ReducerBase

The following table lists the major function APIs.

Major APIs

API	Description
void cleanup(TaskContext context)	The Reduce method is called after the reduce stage ends.
void reduce(Record key, Iterator<Record > values, TaskContext context)	Reduce method, processing records in input table.
void setup(TaskContext context)	The Reduce method is called before the reduce stage begins.

1.8.3.2.3. TaskContext

The following table lists the major function APIs.

Major APIs

API	Description
TableInfo[] getOutputTableInfo()	Get output table information.
Record createOutputRecord()	Create record object of default output table.
Record createOutputRecord(String label)	Create record object of given label output table.
Record createMapOutputKeyRecord()	Create record object of Map output Key.
Record createMapOutputValueRecord()	Create record object of Map output Value.
void write(Record record)	Write record to default output for writing data at Reduce end. Can be called multiple times at Reduce end.
void write(Record record, String label)	Write record to given label output for writing data at Reduce end. Can be called multiple times at Reduce end.
void write(Record key, Record value)	Write record to intermediate result for writing data at Map end. Can be called multiple times at Map end.

API	Description
BufferedInputStream readResourceFileAsStream(String resourceName)	Read file type resource.
Iterator<Record > readResourceTable(String resourceName)	Reads table type resources.
Counter getCounter(Enum<? > name)	Get Counter object of given name.
Counter getCounter(String group, String name)	Get Counter object of given group name and name.
void progress()	Report heartbeat information to the MapReduce framework. If the processing time of your method is extended, and you have not called the framework during the current process, call it to avoid a task timeout period. 600 seconds is set as the timeout period by default.

Note TaskContext API has a progress function which prevents it from being forced out due to time-out if a worker runs for a long time. This API sends heartbeats to the framework rather than reporting the Worker progress. The default worker timeout period for MaxCompute MapReduce is 10 minutes (which cannot be modified by the user). If the worker does not send a heartbeat (calling the progress API) in 10 minutes, the framework terminates the worker and the MapReudce task fails. We recommend that you let the worker call the progress API periodically in the Mapper/Reducer function to prevent the framework from terminating the task unexpectedly.

1.8.3.2.4. JobConf

The following table lists the major function APIs.

Major APIs

API	Description
void setResources(String resourceNames)	Declare resources this job uses. Only declared resources are available to be read through TaskContext object while run Mapper/Reducer.
void setMapOutputKeySchema(Column[] schema)	Set attribute of Key output from Mapper to Reducer.
void setMapOutputValueSchema(Column[] schema)	Set attribute of Value output from Mapper to Reducer.
void setOutputKeySortColumns(String[] cols)	Set sorting columns of Key output from Mapper to Reducer.
void setOutputGroupingColumns(String[] cols)	Set Key grouping columns.

API	Description
void setMapperClass(Class<? extends Mapper > > > theClass)	Set Mapper function of job.
void setPartitionColumns(String[] cols)	Set partition columns designated by job. It is all the columns of Mapper output Key by default.
void setReducerClass(Class<? extends Reducer > theClass)	Set the job reducer.
void setCombinerClass(Class<? extends Reducer > theClass)	Set job combiner. When run at Map side, the function is similar to a single Map and local Key value of the local Reduce.
void setSplitSize(long size)	Set input split size, unit: MB, default value: 640.
void setNumReduceTasks(int n)	Set number of Reducer task, it is 1/4 number of Mapper task by default.
void setMemoryForMapTask(int mem)	Set memory size of single Worker in Mapper tasks, unit: MB, default value: 2048.
void setMemoryForReduceTask(int mem)	Set memory size of single Worker in Reduce tasks, unit: MB, default value: 2048.
void setOutputSchema(Column[] schema, String label)	Set output attribute of designated label. While multiplexed output, each output corresponds to 1 label.

Note

- Normally, GroupingColumns is included in KeySortColumns, and KeySortColumns is included in Key.
- At the Map side, the Record output from Mapper calculates Hash value and then, based on the set PartitionColumns, determines which Reducer to distribute to for sorting Records based on KeySortColumns.
- At the Reduce side, after sorting input Records based on KeySortColumns, Records are grouped based on columns designated by GroupingColumns, and are input in traversal sequence. The use of the same Records in columns designated by GroupingColumns as 1 input called by reduce function.

1.8.3.2.5. JobClient

The following table lists the major function APIs.

Major APIs

API	Description

API	Description
static RunningJob runJob(JobConf job)	Use blocking (synchronous) mode to submit MapReduce jobs and return immediately.
static RunningJob submitJob(JobConf job)	Use non-blocking (asynchronous) mode to submit MapReduce jobs and return immediately.

1.8.3.2.6. RunningJob

The following table lists the major function APIs.

Major APIs

API	Description
String getInstanceID()	Get instance ID for checking run log and job management.
boolean isComplete()	Check whether job is complete.
boolean isSuccessful()	Check whether job instance is successful.
void waitForCompletion()	Wait until job instance is complete. It is typically is used for jobs submitted is asynchronous mode.
JobStatus getJobStatus()	Check job instance status.
void killJob()	End the job.
Counters getCounters()	Get Counter information.

1.8.3.2.7. InputUtils

The following table lists the major function APIs.

Major APIs

API	Description
static void addTable(TableInfo table, JobConf conf)	Add table to task input. It can be called more than once. Newly added tables are appended to input queue.
static void setTables(TableInfo [] tables, JobConf conf)	Add tables to task input.

1.8.3.2.8. OutputUtils

The following table lists the major function APIs.

Major APIs

API	Description
static void addTable(TableInfo table, JobConf conf)	Adds a table to the task output. It can be called more than once. Newly added tables are appended to the output queue.
static void setTables(TableInfo [] tables, JobConf conf)	Adds multiple tables to the task output.

1.8.3.2.9. Pipeline

Pipeline is the main class of MR2. A Pipeline can be built with Pipeline.Builder. Major Pipeline APIs are as follows:

```
public Builder addMapper(Class<? extends Mapper> mapper)
public Builder addMapper(Class<? extends Mapper> mapper,Column[] keySchema, Column[] valueSche
ma, String[] sortCols, SortOrder[] order, String[] partCols,Class<? extends Partitioner> theClass, String
[] groupCols)
public Builder addReducer(Class<? extends Reducer> reducer)
public Builder addReducer(Class<? extends Reducer> reducer,Column[] keySchema, Column[] valueSch
ema, String[] sortCols, SortOrder[] order, String[] partCols,Class<? extends Partitioner> theClass, Strin
g[] groupCols)
public setoutputkeyschema builder (Column [] keyschema)
public setoutputvalueschema builder (Column [] valueschema)
public setoutputkeysortcolumns builder (String [] sortcols)
public setoutputkeysortorder builder (Sortorder [] order)
public setpartitioncolumns builder (String [] partcols)
public Builder setPartitionerClass(Class<? extends Partitioner> theClass)
public Builder setOutputGroupingColumns(String[] cols)
```

Example:

```
job job = new job ();
pipeline pipeline = pipeline. builder ()
. Addmapper (maid. Class)
.setOutputKeySchema(
new Column[] { new Column("word", OdpsType.STRING) })
.setOutputValueSchema(
new Column[] { new Column("count", OdpsType.BIGINT) })
. addreducer (Sumreducer. class)
. setoutputkeyschema (
new Column[] { new Column("count", OdpsType.BIGINT) })
.setOutputValueSchema(
new column [] {new column ("word", OdpsType. string),
new column ("count", OdpsType. bigint)})
. addreducer (Identityreducer. class). createPipeline ();
job.setPipeline(pipeline); job.addInput(...)
job.addOutput(...) job.submit();
```

As shown in the preceding example, you can build MapReduce tasks of a Map operation followed by two Reduce operations in the MAIN function. If you are familiar with the basic features of MapReduce, you can use MR2 easily . We also recommend that you learn the basic features of MapReduce before using MR2, namely configuring MapReduce tasks through JobConf. JobConf can only configure MapReduce tasks of a Map operation followed by a single Reduce operation.

1.8.3.3. Compatibility with Hadoop MapReduce

The following table describes whether specific MaxCompute MapReduce interfaces are compatible with Hadoop MapReduce.

Type	Interface	Compatible with Hadoop MapReduce?
Mapper	void map(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Mapper.Context context)	Yes
Mapper	void run(org.apache.hadoop.mapreduce.Mapper.Context context)	Yes
Mapper	void setup(org.apache.hadoop.mapreduce.Mapper.Context context)	Yes

Type	Interface	Compatible with Hadoop MapReduce?
Reducer	void cleanup(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes
Reducer	void reduce(KEYIN key, VALUEIN value, org.apache.hadoop.mapreduce.Reducer.Context context)	Yes
Reducer	void run(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes
Reducer	void setup(org.apache.hadoop.mapreduce.Reducer.Context context)	Yes
Partitioner	int getPartition(KEY key, VALUE value, int numPartitions)	Yes
MapContext (which extends TaskInputOutputContext)	InputSplit getInputSplit()	No. An exception is reported.
ReduceContext	nextKey()	Yes
ReduceContext	getValues()	Yes
TaskInputOutputContext	getCurrentKey()	Yes
TaskInputOutputContext	getCurrentValue()	Yes
TaskInputOutputContext	getOutputCommitter()	No. An exception is reported.
TaskInputOutputContext	nextKeyValue()	Yes
TaskInputOutputContext	write(KEYOUT key, VALUEOUT value)	Yes
TaskAttemptContext	getCounter(Enum<? > counterName)	Yes
TaskAttemptContext	getCounter(String groupName, String counterName)	Yes
TaskAttemptContext	setStatus(String msg)	Empty implementation
TaskAttemptContext	getStatus()	Empty implementation
TaskAttemptContext	getTaskAttemptID()	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
TaskAttemptContext	getProgress()	No. An exception is reported.
TaskAttemptContext	progress()	Yes
Job	addArchiveToClassPath(Path archive)	No
Job	addCacheArchive(URI uri)	No
Job	addCacheFile(URI uri)	No
Job	addFileToClassPath(Path file)	No
Job	cleanupProgress()	No
Job	createSymlink()	No. An exception is reported.
Job	failTask(TaskAttemptID taskId)	No
Job	getCompletionPollInterval(Configuration conf)	Empty implementation
Job	getCounters()	Yes
Job	getFinishTime()	Yes
Job	getHistoryUrl()	Yes
Job	getInstance()	Yes
Job	getInstance(Cluster ignored)	Yes
Job	getInstance(Cluster ignored, Configuration conf)	Yes
Job	getInstance(Configuration conf)	Yes
Job	getInstance(Configuration conf, String jobName)	Empty implementation
Job	getInstance(JobStatus status, Configuration conf)	No. An exception is reported.
Job	getJobFile()	No. An exception is reported.
Job	getJobName()	Empty implementation
Job	getJobState()	No. An exception is reported.
Job	getPriority()	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
Job	getProgressPollInterval(Configuration conf)	Empty implementation
Job	getReservationId()	No. An exception is reported.
Job	getSchedulingInfo()	No. An exception is reported.
Job	getStartTime()	Yes
Job	getStatus()	No. An exception is reported.
Job	getTaskCompletionEvents(int startFrom)	No. An exception is reported.
Job	getTaskCompletionEvents(int startFrom, int numEvents)	No. An exception is reported.
Job	getTaskDiagnostics(TaskAttemptID taskid)	No. An exception is reported.
Job	getTaskOutputFilter(Configuration conf)	No. An exception is reported.
Job	getTaskReports(TaskType type)	No. An exception is reported.
Job	getTrackingURL()	Yes
Job	isComplete()	Yes
Job	isRetired()	No. An exception is reported.
Job	isSuccessful()	Yes
Job	isUber()	Empty implementation
Job	killJob()	Yes
Job	killTask(TaskAttemptID taskId)	No
Job	mapProgress()	Yes
Job	monitorAndPrintJob()	Yes
Job	reduceProgress()	Yes
Job	setCacheArchives(URI[] archives)	No. An exception is reported.
Job	setCacheFiles(URI[] files)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
Job	setCancelDelegationTokenUponJobCompletion(boolean value)	No. An exception is reported.
Job	setCombinerClass(Class<? extends Reducer> cls)	Yes
Job	setCombinerKeyGroupingComparatorClass(Class<? extends RawComparator> cls)	Yes
Job	setGroupingComparatorClass(Class<? extends RawComparator> cls)	Yes
Job	setInputFormatClass(Class<? extends InputFormat> cls)	Empty implementation
Job	setJar(String jar)	Yes
Job	setJarByClass(Class<? > cls)	Yes
Job	setJobName(String name)	Empty implementation
Job	setJobSetupCleanupNeeded(boolean needed)	Empty implementation
Job	setMapOutputKeyClass(Class<? > theClass)	Yes
Job	setMapOutputValueClass(Class<? > theClass)	Yes
Job	setMapperClass(Class<? extends Mapper> cls)	Yes
Job	setMapSpeculativeExecution(boolean speculativeExecution)	Empty implementation
Job	setMaxMapAttempts(int n)	Empty implementation
Job	setMaxReduceAttempts(int n)	Empty implementation
Job	setNumReduceTasks(int tasks)	Yes
Job	setOutputFormatClass(Class<? extends OutputFormat> cls)	No. An exception is reported.
Job	setOutputKeyClass(Class<? > theClass)	Yes

Type	Interface	Compatible with Hadoop MapReduce?
Job	setOutputValueClass(Class<? > theClass)	Yes
Job	setPartitionerClass(Class<? extends Partitioner> cls)	Yes
Job	setPriority(JobPriority priority)	No. An exception is reported.
Job	setProfileEnabled(boolean newValue)	Empty implementation
Job	setProfileParams(String value)	Empty implementation
Job	setProfileTaskRange(boolean isMap, String newValue)	Empty implementation
Job	setReducerClass(Class<? extends Reducer> cls)	Yes
Job	setReduceSpeculativeExecution(boolean speculativeExecution)	Empty implementation
Job	setReservationId(ReservationId reservationId)	No. An exception is reported.
Job	setSortComparatorClass(Class <? extends RawComparator> cls)	No. An exception is reported.
Job	setSpeculativeExecution(boolean speculativeExecution)	Yes
Job	setTaskOutputFilter(Configuration conf, org.apache.hadoop.mapreduce.Job.TaskStatusFilter newValue)	No. An exception is reported.
Job	setupProgress()	No. An exception is reported.
Job	setUser(String user)	Empty implementation
Job	setWorkingDirectory(Path dir)	Empty implementation
Job	submit()	Yes
Job	toString()	No. An exception is reported.
Job	waitForCompletion(boolean verbose)	Yes

Type	Interface	Compatible with Hadoop MapReduce?
Task Execution & Environment	mapreduce.map.java.opts	Empty implementation
Task Execution & Environment	mapreduce.reduce.java.opts	Empty implementation
Task Execution & Environment	mapreduce.map.memory.mb	Empty implementation
Task Execution & Environment	mapreduce.reduce.memory.mb	Empty implementation
Task Execution & Environment	mapreduce.task.io.sort.mb	Empty implementation
Task Execution & Environment	mapreduce.map.sort.spill.percent	Empty implementation
Task Execution & Environment	mapreduce.task.io.soft.factor	Empty implementation
Task Execution & Environment	mapreduce.reduce.merge.inmem.thresholds	Empty implementation
Task Execution & Environment	mapreduce.reduce.shuffle.merge.percent	Empty implementation
Task Execution & Environment	mapreduce.reduce.shuffle.input.buffer.percent	Empty implementation
Task Execution & Environment	mapreduce.reduce.input.buffer.percent	Empty implementation
Task Execution & Environment	mapreduce.job.id	Empty implementation
Task Execution & Environment	mapreduce.job.jar	Empty implementation
Task Execution & Environment	mapreduce.job.local.dir	Empty implementation
Task Execution & Environment	mapreduce.task.id	Empty implementation
Task Execution & Environment	mapreduce.task.attempt.id	Empty implementation
Task Execution & Environment	mapreduce.task.is.map	Empty implementation
Task Execution & Environment	mapreduce.task.partition	Empty implementation
Task Execution & Environment	mapreduce.map.input.file	Empty implementation
Task Execution & Environment	mapreduce.map.input.start	Empty implementation
Task Execution & Environment	mapreduce.map.input.length	Empty implementation
Task Execution & Environment	mapreduce.task.output.dir	Empty implementation

Type	Interface	Compatible with Hadoop MapReduce?
JobClient	cancelDelegationToken(Token <DelegationTokenIdentifier> token)	No. An exception is reported.
JobClient	close()	Empty implementation
JobClient	displayTasks(JobID jobId, String type, String state)	No. An exception is reported.
JobClient	getAllJobs()	No. An exception is reported.
JobClient	getCleanupTaskReports(JobID jobId)	No. An exception is reported.
JobClient	getClusterStatus()	No. An exception is reported.
JobClient	getClusterStatus(boolean detailed)	No. An exception is reported.
JobClient	getDefaultMaps()	No. An exception is reported.
JobClient	getDefaultReduces()	No. An exception is reported.
JobClient	getDelegationToken(Text renewer)	No. An exception is reported.
JobClient	getFs()	No. An exception is reported.
JobClient	getJob(JobID jobid)	No. An exception is reported.
JobClient	getJob(String jobid)	No. An exception is reported.
JobClient	getJobsFromQueue(String queueName)	No. An exception is reported.
JobClient	getMapTaskReports(JobID jobId)	No. An exception is reported.
JobClient	getMapTaskReports(String jobId)	No. An exception is reported.
JobClient	getQueueAclsForCurrentUser()	No. An exception is reported.
JobClient	getQueueInfo(String queueName)	No. An exception is reported.
JobClient	getQueues()	No. An exception is reported.
JobClient	getReduceTaskReports(JobID jobId)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
JobClient	getReduceTaskReports(String jobId)	No. An exception is reported.
JobClient	getSetupTaskReports(JobID jobId)	No. An exception is reported.
JobClient	getStagingAreaDir()	No. An exception is reported.
JobClient	getSystemDir()	No. An exception is reported.
JobClient	getTaskOutputFilter()	No. An exception is reported.
JobClient	getTaskOutputFilter(JobConf job)	No. An exception is reported.
JobClient	init(JobConf conf)	No. An exception is reported.
JobClient	isJobDirValid(Path jobDirPath, FileSystem fs)	No. An exception is reported.
JobClient	jobsToComplete()	No. An exception is reported.
JobClient	monitorAndPrintJob(JobConf conf, RunningJob job)	No. An exception is reported.
JobClient	renewDelegationToken(Token <DelegationTokenIdentifier> token)	No. An exception is reported.
JobClient	run(String[] argv)	No. An exception is reported.
JobClient	runJob(JobConf job)	Yes
JobClient	setTaskOutputFilter(JobClient.TaskStatusFilter newValue)	No. An exception is reported.
JobClient	setTaskOutputFilter(JobConf job, JobClient.TaskStatusFilter newValue)	No. An exception is reported.
JobClient	submitJob(JobConf job)	Yes
JobClient	submitJob(String jobFile)	No. An exception is reported.
JobConf	deleteLocalFiles()	No. An exception is reported.
JobConf	deleteLocalFiles(String subdir)	No. An exception is reported.
JobConf	normalizeMemoryConfigValue(long val)	Empty implementation

Type	Interface	Compatible with Hadoop MapReduce?
JobConf	setCombinerClass(Class<? extends Reducer> theClass)	Yes
JobConf	setCompressMapOutput(boolean compress)	Empty implementation
JobConf	setInputFormat(Class<? extends InputFormat> theClass)	No. An exception is reported.
JobConf	setJar(String jar)	No. An exception is reported.
JobConf	setJarByClass(Class cls)	No. An exception is reported.
JobConf	setJobEndNotificationURI(String uri)	No. An exception is reported.
JobConf	setJobName(String name)	Empty implementation
JobConf	setJobPriority(JobPriority prio)	No. An exception is reported.
JobConf	setKeepFailedTaskFiles(boolean keep)	No. An exception is reported.
JobConf	setKeepTaskFilesPattern(String pattern)	No. An exception is reported.
JobConf	setKeyFieldComparatorOptions(String keySpec)	No. An exception is reported.
JobConf	setKeyFieldPartitionerOptions(String keySpec)	No. An exception is reported.
JobConf	setMapDebugScript(String mDbgScript)	Empty implementation
JobConf	setMapOutputCompressorClass(Class<? extends CompressionCodec> codecClass)	Empty implementation
JobConf	setMapOutputKeyClass(Class<? > theClass)	Yes
JobConf	setMapOutputValueClass(Class<? > theClass)	Yes
JobConf	setMapperClass(Class<? extends Mapper> theClass)	Yes

Type	Interface	Compatible with Hadoop MapReduce?
JobConf	setMapRunnerClass(Class<? extends MapRunnable> theClass)	No. An exception is reported.
JobConf	setMapSpeculativeExecution(boolean speculativeExecution)	Empty implementation
JobConf	setMaxMapAttempts(int n)	Empty implementation
JobConf	setMaxMapTaskFailuresPercent(int percent)	Empty implementation
JobConf	setMaxPhysicalMemoryForTask(long mem)	Empty implementation
JobConf	setMaxReduceAttempts(int n)	Empty implementation
JobConf	setMaxReduceTaskFailuresPercent(int percent)	Empty implementation
JobConf	setMaxTaskFailuresPerTracker(int noFailures)	Empty implementation
JobConf	setMaxVirtualMemoryForTask(long vmem)	Empty implementation
JobConf	setMemoryForMapTask(long mem)	Yes
JobConf	setMemoryForReduceTask(long mem)	Yes
JobConf	setNumMapTasks(int n)	Yes
JobConf	setNumReduceTasks(int n)	Yes
JobConf	setNumTasksToExecutePerJvm(int numTasks)	Empty implementation
JobConf	setOutputCommitter(Class<? extends OutputCommitter> theClass)	No. An exception is reported.
JobConf	setOutputFormat(Class<? extends OutputFormat> theClass)	Empty implementation
JobConf	setOutputKeyClass(Class<? > theClass)	Yes

Type	Interface	Compatible with Hadoop MapReduce?
JobConf	setOutputKeyComparatorClass (Class<? extends RawComparator> theClass)	No. An exception is reported.
JobConf	setOutputValueClass(Class<? > theClass)	Yes
JobConf	setOutputValueGroupingComparator(Class<? extends RawComparator> theClass)	No. An exception is reported.
JobConf	setPartitionerClass(Class<? extends Partitioner> theClass)	Yes
JobConf	setProfileEnabled(boolean newValue)	Empty implementation
JobConf	setProfileParams(String value)	Empty implementation
JobConf	setProfileTaskRange(boolean isMap, String newValue)	Empty implementation
JobConf	setQueueName(String queueName)	No. An exception is reported.
JobConf	setReduceDebugScript(String rDbgScript)	Empty implementation
JobConf	setReducerClass(Class<? extends Reducer> theClass)	Yes
JobConf	setReduceSpeculativeExecution(boolean speculativeExecution)	Empty implementation
JobConf	setSessionId(String sessionId)	Empty implementation
JobConf	setSpeculativeExecution(boolean speculativeExecution)	No. An exception is reported.
JobConf	setUseNewMapper(boolean flag)	Yes
JobConf	setUseNewReducer(boolean flag)	Yes
JobConf	setUser(String user)	Empty implementation
JobConf	setWorkingDirectory(Path dir)	Empty implementation
FileInputFormat	N/A	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
TextInputFormat	N/A	Yes
InputSplit	mapred.min.split.size.	No. An exception is reported.
FileSplit	map.input.file	No. An exception is reported.
RecordWriter	N/A	No. An exception is reported.
RecordReader	N/A	No. An exception is reported.
OutputFormat	N/A	No. An exception is reported.
OutputCommitter	abortJob(JobContext jobContext, int status)	No. An exception is reported.
OutputCommitter	abortJob(JobContext context, JobStatus.State runState)	No. An exception is reported.
OutputCommitter	abortTask(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	abortTask(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	cleanupJob(JobContext jobContext)	No. An exception is reported.
OutputCommitter	cleanupJob(JobContext context)	No. An exception is reported.
OutputCommitter	commitJob(JobContext jobContext)	No. An exception is reported.
OutputCommitter	commitJob(JobContext context)	No. An exception is reported.
OutputCommitter	commitTask(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	needsTaskCommit(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	needsTaskCommit(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	setupJob(JobContext jobContext)	No. An exception is reported.
OutputCommitter	setupJob(JobContext jobContext)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
OutputCommitter	setupTask(TaskAttemptContext taskContext)	No. An exception is reported.
OutputCommitter	setupTask(TaskAttemptContext taskContext)	No. An exception is reported.
Counter	getDisplayName()	Yes
Counter	getName()	Yes
Counter	getValue()	Yes
Counter	increment(long incr)	Yes
Counter	setValue(long value)	Yes
Counter	setDisplayName(String displayName)	Yes
DistributedCache	CACHE_ARCHIVES	No. An exception is reported.
DistributedCache	CACHE_ARCHIVES_SIZES	No. An exception is reported.
DistributedCache	CACHE_ARCHIVES_TIMESTAMPS	No. An exception is reported.
DistributedCache	CACHE_FILES	No. An exception is reported.
DistributedCache	CACHE_FILES_SIZES	No. An exception is reported.
DistributedCache	CACHE_FILES_TIMESTAMPS	No. An exception is reported.
DistributedCache	CACHE_LOCALARCHIVES	No. An exception is reported.
DistributedCache	CACHE_LOCALFILES	No. An exception is reported.
DistributedCache	CACHE_SYMLINK	No. An exception is reported.
DistributedCache	addArchiveToClassPath(Path archive, Configuration conf)	No. An exception is reported.
DistributedCache	addArchiveToClassPath(Path archive, Configuration conf, FileSystem fs)	No. An exception is reported.
DistributedCache	addCacheArchive(URI uri, Configuration conf)	No. An exception is reported.
DistributedCache	addCacheFile(URI uri, Configuration conf)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
DistributedCache	addFileToClassPath(Path file, Configuration conf)	No. An exception is reported.
DistributedCache	addFileToClassPath(Path file, Configuration conf, FileSystem fs)	No. An exception is reported.
DistributedCache	addLocalArchives(Configuration conf, String str)	No. An exception is reported.
DistributedCache	addLocalFiles(Configuration conf, String str)	No. An exception is reported.
DistributedCache	checkURIs(URI[] uriFiles, URI[] uriArchives)	No. An exception is reported.
DistributedCache	createAllSymlink(Configuration conf, File jobCacheDir, File workDir)	No. An exception is reported.
DistributedCache	createSymlink(Configuration conf)	No. An exception is reported.
DistributedCache	getArchiveClassPaths(Configuration conf)	No. An exception is reported.
DistributedCache	getArchiveTimestamps(Configuration conf)	No. An exception is reported.
DistributedCache	getCacheArchives(Configuration conf)	No. An exception is reported.
DistributedCache	getCacheFiles(Configuration conf)	No. An exception is reported.
DistributedCache	getFileClassPaths(Configuration conf)	No. An exception is reported.
DistributedCache	getFileStatus(Configuration conf, URI cache)	No. An exception is reported.
DistributedCache	getFileTimestamps(Configuration conf)	No. An exception is reported.
DistributedCache	getLocalCacheArchives(Configuration conf)	No. An exception is reported.
DistributedCache	getLocalCacheFiles(Configuration conf)	No. An exception is reported.
DistributedCache	getSymlink(Configuration conf)	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
DistributedCache	getTimestamp(Configuration conf, URI cache)	No. An exception is reported.
DistributedCache	setArchiveTimestamps(Configuration conf, String timestamps)	No. An exception is reported.
DistributedCache	setCacheArchives(URI[] archives, Configuration conf)	No. An exception is reported.
DistributedCache	setCacheFiles(URI[] files, Configuration conf)	No. An exception is reported.
DistributedCache	setFileTimestamps(Configuration conf, String timestamps)	No. An exception is reported.
DistributedCache	setLocalArchives(Configuration conf, String str)	No. An exception is reported.
DistributedCache	setLocalFiles(Configuration conf, String str)	No. An exception is reported.
IsolationRunner	N/A	No. An exception is reported.
Profiling	N/A	Empty implementation
Debugging	N/A	Empty implementation
Data Compression	N/A	Yes
Skipping Bad Records	N/A	No. An exception is reported.
Job Authorization	mapred.acls.enabled	No. An exception is reported.
Job Authorization	mapreduce.job.acl-view-job	No. An exception is reported.
Job Authorization	mapreduce.job.acl-modify-job	No. An exception is reported.
Job Authorization	mapreduce.cluster.administrators	No. An exception is reported.
Job Authorization	mapred.queue.queue-name.acl-administer-jobs	No. An exception is reported.
MultipleInputs	N/A	No. An exception is reported.
Multi{anchor:_GoBack}pleOutputs	N/A	Yes
org.apache.hadoop.mapreduce.lib.db	N/A	No. An exception is reported.

Type	Interface	Compatible with Hadoop MapReduce?
org.apache.hadoop.mapreduce.security	N/A	No. An exception is reported.
org.apache.hadoop.mapreduce.lib.jobcontrol	N/A	No. An exception is reported.
org.apache.hadoop.mapreduce.lib.chain	N/A	No. An exception is reported.
org.apache.hadoop.mapreduce.lib.db	N/A	No. An exception is reported.

1.8.4. Data types

MapReduce supports the following data types: bigint, double, string, datetime, boolean, decimal, tinyint, smallint, int, float, varchar, timestamp, binary, array, map, and struct. The following table lists the mappings between MaxCompute data types and Java types.

Data type mapping

MaxCompute type	Java type
Tinyint	java.lang.Byte
Smallint	java.lang.Short
Int	java.lang.Integer
Bigint	java.lang.Long
Float	java.lang.Float
Double	java.lang.Double
Decimal	java.math.BigDecimal
Boolean	java.lang.Boolean
String	java.lang.String
Varchar	com.aliyun.odps.data.Varchar
Binary	com.aliyun.odps.data.Binary
Datetime	java.util.Date
Timestamp	java.sql.Timestamp
Array	java.util.List

MaxCompute type	Java type
Map	java.util.Map
Struct	com.aliyun.odps.data.Struct

1.8.5. Limits

The following limits apply to MapReduce:

- A Map or Reduce worker consumes 2,048 MB memory by default. The value range is 256 MB to 12 GB.
- Each task can reference up to 256 resources. Each partitioned table is considered as one unit.
- Each task can have up to 1,024 inputs and up to 256 outputs.
- Each task can have up to 64 custom counters.
- The number of Map instances in a job is calculated by the framework based on the split size. If no input table is specified, you can use odps.stage.mapper.num to set the number of Map instances. The number is in the range of 1 to 100,000.
- The default number of Reduce instances in a job is 1/4 of the number of Map instances. You can set this number in the range of 0 to 2,000. The following situation may occur: Reduce instances process much more data than Map instances, which results in a slow Reduce stage. Furthermore, a maximum of 2,000 Reduce instances can be created.
- Each Map or Reduce instance can retry three times after failure. Exceptions that do not allow retries can cause a job to fail.
- For local jobs, the number of Map or Reduce workers cannot exceed 100, while the number of downloads for an input is 100 by default.
- Each Map or Reduce worker can read a resource a maximum of 64 times.
- A task can reference a maximum of 2 GB resources.
- The framework determines the number of Map instances based on the split size.
- The length of string columns in MaxCompute tables cannot exceed 8 MB.
- If no data access operations or heartbeat packets are sent through context.progress(), the default timeout period of a Map or Reduce worker is 600s.

1.8.6. Sample programs

1.8.6.1. WordCount example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException; import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
```

```
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
public class WordCount {
public static class TokenizerMapper extends MapperBase {
private Record word;
private Record one;
@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.set(new Object[] { 1L });
System.out.println("TaskID:" + context.getTaskID().toString());
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
for (int i = 0; i < record.getColumnCount(); i++) {
word.set(new Object[] { record.get(i).toString() });
context.write(word, one);
}
}
}
/**
*A combiner class that combines map output by sum them.
**/
public static class SumCombiner extends ReducerBase {
private Record count;
@Override
public void setup(TaskContext context) throws IOException {
count = context.createMapOutputValueRecord();
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context)
throws IOException {
long c = 0;
while (values.hasNext()) {
Record val = values.next();
c += (Long) val.get(0);
}
```

```
count.set(0, c);
context.write(key, count);
}
}
/**
* A reducer class that just emits the sum of the input values.
**/
public static class SumReducer extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException { result = context.createOutputRecord();
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context)
throws IOException {
long count = 0;
while (values.hasNext()) {
Record val = values.next();
count += (Long) val.get(0);
}
result.set(0, key.get(0));
result.set(1, count);
context.write(result);
}
}
public static void main(String[] args)
throws Exception {
if (args.length ! = 2) {
System.err.println("Usage: WordCount <in_table> <out_table>");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(SumCombiner.class);
job.setReducerClass(SumReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
```

```
}
```

1.8.6.2. MapOnly example

For MapOnly jobs, Map directly sends < Key, Value > pairs to tables on MaxCompute. You only need to specify output tables. You do not need to specify the key-value metadata for map output.

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
public class MapOnly {
public static class MapperClass extends MapperBase {
@Override
public void setup(TaskContext context)
throws IOException {
boolean is = context.getJobConf().getBoolean("option.mapper.setup", false);
if (is) {
Record result = context.createOutputRecord();
result.set(0, "setup");
result.set(1, 1L); context.write(result);
}
}
@Override
public void map(long key, Record record, TaskContext context) throws IOException {
boolean is = context.getJobConf().getBoolean("option.mapper.map", false);
if (is) {
Record result = context.createOutputRecord();
result.set(0, record.get(0));
result.set(1, 1L); context.write(result);
}
}
@Override
```

```
public void cleanup(TaskContext context) throws IOException {
boolean is = context.getJobConf().getBoolean("option.mapper.cleanup", false);
if (is) {
Record result = context.createOutputRecord();
result.set(0, "cleanup");
result.set(1, 1L); context.write(result);
}
}
}
public static void main(String[] args) throws Exception {
if (args.length ! = 2 && args.length ! = 3) {
System.err.println("Usage: OnlyMapper <in_table> <out_table> [setup|map|cleanup]");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(MapperClass.class);
job.setNumReduceTasks(0);
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
if (args.length == 3) {
String options = new String(args[2]);
if (options.contains("setup")) {
job.setBoolean("option.mapper.setup", true);
}
if (options.contains("map")) {
job.setBoolean("option.mapper.map", true);
}
if (options.contains("cleanup")) {
job.setBoolean("option.mapper.cleanup", true);
}
}
JobClient.runJob(job);
}
}
```

1.8.6.3. Example: Input and output data to multiple objects

MaxCompute jobs can read data from multiple input tables, and write data to multiple output tables. All input tables for a job must have the same schema (number and type of columns). The output tables for a job can have different schemas (number and type of columns).

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException; import java.util.Iterator;
import java.util.LinkedHashMap;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
*Multi input & output example.
*
*To run: jar -resources odps-mapred-example-0.12.0.jar com.aliyun.odps.mapred.open.example.Multipl
eInOut
*mr_src,mr_src1,mr_srcpart|pt=1,mr_srcpart|pt=2/ds=2
*mr_multiinout_out1,mr_multiinout_out2|a=1/b=1|out1,mr_multiinout_out2|a=2/b=2|out2;
*
**/
public class MultipleInOut {
public static class TokenizerMapper extends MapperBase {
private Record word;
private Record one;
@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.set(new Object[] { 1L });
}
@Override
public void map(long recordNum, Record record, TaskContext context)
throws IOException {
for (int i = 0; i < record.getColumnCount(); i++) {
word.set(new Object[] { record.get(i).toString() });
```

```
word.set(new Object[] { record.get(i).toString() });
context.write(word, one);
}
}
}
public static class SumReducer extends ReducerBase {
private Record result;
private Record result1;
private Record result2;
@Override
public void setup(TaskContext context) throws IOException {
result = context.createOutputRecord();
result1 = context.createOutputRecord("out1");
result2 = context.createOutputRecord("out2");
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context)
throws IOException { long count = 0;
while (values.hasNext()) {
Record val = values.next();
count += (Long) val.get(0);
}
long mod = count % 3;
if (mod == 0) {
result.set(0, key.get(0));
result.set(1, count);
// No label is specified. Default output is adopted.
context.write(result);
} else if (mod == 1) {
result1.set(0, key.get(0));
result1.set(1, count);
context.write(result1, "out1");
} else {
result2.set(0, key.get(0));
result2.set(1, count);
context.write(result2, "out2");
}
}
@Override
public void cleanup(TaskContext context) throws IOException {
Record result = context.createOutputRecord();
```

```
result.set(0, "default");
result.set(1, 1L);
context.write(result);
Record result1 = context.createOutputRecord("out1");
result1.set(0, "out1");
result1.set(1, 1L); context.write(result1, "out1");
Record result2 = context.createOutputRecord("out2");
result2.set(0, "out1");
result2.set(1, 1L); context.write(result2, "out2");
}
}
public static LinkedHashMap<String, String> convertPartSpecToMap( String partSpec) {
LinkedHashMap<String, String> map = new LinkedHashMap<String, String>();
if (partSpec ! = null && ! partSpec.trim().isEmpty()) {
String[] parts = partSpec.split("/");
for (String part : parts) {
String[] ss = part.split("=");
if (ss.length ! = 2) {
throw new RuntimeException("ODPS-0730001: error part spec format: "+ partSpec);
}
map.put(ss[0], ss[1]);
}
}
return map;
}
public static void main(String[] args) throws Exception {
String[] inputs = null;
String[] outputs = null;
if (args.length == 2) {
inputs = args[0].split(",");
outputs = args[1].split(",");
} else {
System.err.println("MultipleInOut in... out...");
System.exit(1);
}
JobConf job = new JobConf();
job.setMapperClass(TokenizerMapper.class);
job.setReducerClass(SumReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
 // Parse input table strings.
```

```
for (String in : inputs) { String[] ss = in.split("\\|");
if (ss.length == 1) {
InputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
} else if (ss.length == 2) {
LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
InputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
} else {
System.err.println("Style of input: " + in + " is not right");
System.exit(1);
}
}
// Parse output table strings.
for (String out : outputs) { String[] ss = out.split("\\|");
if (ss.length == 1) {
OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).build(), job);
} else if (ss.length == 2) {
LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
} else if (ss.length == 3) {
if (ss[1].isEmpty()) {
LinkedHashMap<String, String> map = convertPartSpecToMap(ss[2]);
OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).build(), job);
} else {
LinkedHashMap<String, String> map = convertPartSpecToMap(ss[1]);
OutputUtils.addTable(TableInfo.builder().tableName(ss[0]).partSpec(map).label(ss[2]).build(), job);
}} else {
System.err.println("Style of output: " + out + " is not right");
System.exit(1);
}
}
JobClient.runJob(job);
}
}
```

1.8.6.4. Multi-task example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
```

```
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* MultiJobs
*
* Running multiple job
*
* To run: jar -resources > multijobs_res_table,odps-mapred-example-0.12.0.jar
* com.aliyun.odps.mapred.open.example.MultiJobs mr_multijobs_out;
*
**/
public class MultiJobs {
public static class InitMapper extends MapperBase {
@Override
public void setup(TaskContext context) throws IOException { Record record = context.createOutputRec
ord();
long v = context.getJobConf().getLong("multijobs.value", 2);
record.set(0, v);
context.write(record);
}
}
public static class DecreaseMapper extends MapperBase {
@Override
public void cleanup(TaskContext context) throws IOException {
 // Obtain the variable values defined by the main function from JobConf.
long expect = context.getJobConf().getLong("multijobs.expect.value", -1);
long v = -1;
int count = 0;
Iterator<Record> iter = context.readResourceTable("multijobs_res_table");
while (iter.hasNext()) {
Record r = iter.next();
v = (Long) r.get(0);
if (expect ! = v) {
```

```
throw new IOException("expect: " + expect + ", but: " + v);
}
count++;
}
if (count ! = 1) {
throw new IOException("res_table should have 1 record, but: " + count);
}
Record record = context.createOutputRecord();
v--;
record.set(0, v);
context.write(record);
context.getCounter("multijobs", "value").setValue(v);
}
}
public static void main(String[] args) throws Exception { if (args.length ! = 1) {
System.err.println("Usage: TestMultiJobs <table>");
System.exit(1);
}
String tbl = args[0];
long iterCount = 2;
System.err.println("Start to run init job.");
JobConf initJob = new JobConf();
initJob.setLong("multijobs.value", iterCount);
initJob.setMapperClass(InitMapper.class);
InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), initJob);
OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), initJob);
initJob.setMapOutputKeySchema(SchemaUtils.fromString("key:string"));
initJob.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));
initJob.setNumReduceTasks(0); JobClient.runJob(initJob);
while (true) {
System.err.println("Start to run iter job, count: " + iterCount);
JobConf decJob = new JobConf();
decJob.setLong("multijobs.expect.value", iterCount);
decJob.setMapperClass(DecreaseMapper.class);
InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), decJob);
OutputUtils.addTable(TableInfo.builder().tableName(tbl).build(), decJob);
decJob.setNumReduceTasks(0);
RunningJob rJob = JobClient.runJob(decJob); iterCount--;
if (rJob.getCounters().findCounter("multijobs", "value").getValue() == 0) { break;
}
```

```
}
if (iterCount ! = 0) {
throw new IOException("Job failed.");
}
}
}
```

1.8.6.5. Secondary sorting example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
/**
* This is an example ODPS Map/Reduce application. It reads the input > table that
* must contain two integers per record. The output is sorted by the > first and
* second number and grouped by the first number.
*
* To run: jar -resources odps-mapred-example-0.12.0.jar
* com.aliyun.odps.mapred.open.example.SecondarySort mr_sort_in > mr_secondarysort_out;
*
**/
public class SecondarySort {
/**
* Read two integers from each line and generate a key, value pair as > ((left,
* right), right).
**/
public static class MapClass extends MapperBase { private Record key;
private Record value;
@Override
public void setup(TaskContext context) throws IOException { key = context.createMapOutputKeyRecor
```

```
public void setup(TaskContext context) throws IOException { key = context.createMapOutputKeyRecor
d();
value = context.createMapOutputValueRecord();
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
long left = 0;
long right = 0;
if (record.getColumnCount() > 0) { left = (Long) record.get(0);
if (record.getColumnCount() > 1) { right = (Long) record.get(1);
}
key.set(new Object[] { (Long) left, (Long) right });
value.set(new Object[] { (Long) right });
context.write(key, value);
}
}
}
/**
* A reducer class that just emits the sum of the input values.
**/
public static class ReduceClass extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException { result = context.createOutputRecord();
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
result.set(0, key.get(0));
while (values.hasNext()) {
Record value = values.next();
result.set(1, value.get(0));
context.write(result);
}
}
}
public static void main(String[] args) throws Exception {
if (args.length ! = 2) {
System.err.println("Usage: secondarysrot <in> <out>");
System.exit(2);
}
JobConf job = new JobConf();
```

```
job.setMapperClass(MapClass.class);
job.setReducerClass(ReduceClass.class);
// Set multiple columns as keys.
// compare first and second parts of the pair
job.setOutputKeySortColumns(new String[] { "i1", "i2" });
// partition based on the first part of the pair
job.setPartitionColumns(new String[] { "i1" });
// grouping comparator based on the first part of the pair
job.setOutputGroupingColumns(new String[] { "i1" });
// the map output is LongPair, Long
job.setMapOutputKeySchema(SchemaUtils.fromString("i1:bigint,i2:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("i2x:bigint"));
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job); System.exit(0);
}
}
```

1.8.6.6. Resource usage example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.BufferedInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* Upload
*
* Import data from text file into table
*
* To run: jar -resources > odps-mapred-example-0.12.0.jar,mr_join_src1.txt
* com.aliyun.odps.mapred.open.example.Upload mr_join_src1.txt > mr_join_src1;
```

```
* com.aliyun.odps.mapred.open.example.Upload mr_join_src1.txt > mr_join_src1;
*
**/
public class Upload {
public static class UploadMapper extends MapperBase {
@Override
public void setup(TaskContext context) throws IOException { Record record = context.createOutputRec
ord();
StringBuilder importdata = new StringBuilder();
BufferedInputStream bufferedInput = null;
try {
byte[] buffer = new byte[1024]; int bytesRead = 0;
String filename = context.getJobConf().get("import.filename");
bufferedInput = context.readResourceFileAsStream(filename);
while ((bytesRead = bufferedInput.read(buffer)) ! = -1) {
String chunk = new String(buffer, 0, bytesRead);
importdata.append(chunk);
}
String lines[] = importdata.toString().split("\n"); for (int i = 0; i < lines.length; i++) {
String[] ss = lines[i].split(",");
record.set(0, Long.parseLong(ss[0].trim()));
record.set(1, ss[1].trim()); context.write(record);
}
} catch (FileNotFoundException ex) { throw new IOException(ex);
} catch (IOException ex) { throw new IOException(ex);
} finally {
}
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
}
}
public static void main(String[] args) throws Exception { if (args.length ! = 2) {
System.err.println("Usage: Upload <import_txt> <out_table>");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(UploadMapper.class);
job.set("import.filename", args[0]);
job.setNumReduceTasks(0);
job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint"));
```

```
job.setMapOutputValueSchema(SchemaUtils.fromString("value:string"));
InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
}
```

> **Note**
>
> You can use the following methods to configure JobConf:
>
> - Use the JobConf API in the SDK. This method has the highest priority.
> - Use the -conf parameter in a jar command to specify a new JobConf file. This method has the lowest priority. For how to use -Conf, refer to the relevant command.

1.8.6.7. Example for using counters

Three counters are defined in this example: map_outputs, reduce_outputs, and global_counts. You can use the setup, map, reduce and cleanup APIs of the Map or Reduce function to obtain and operate custom counters.

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException; import java.util.Iterator;
import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.counter.Counters;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.SchemaUtils;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.data.TableInfo;
/**
* User Defined Counters
*
* To run: jar -resources odps-mapred-example-0.12.0.jar
* com.aliyun.odps.mapred.open.example.UserDefinedCounters mr_src > mr_testcounters_out;
*
**/
```

```
public class UserDefinedCounters {
enum MyCounter {
TOTAL_TASKS, MAP_TASKS, REDUCE_TASKS
}
public static class TokenizerMapper extends MapperBase {
private Record word;
private Record one;
@Override
public void setup(TaskContext context) throws IOException { super.setup(context);
Counter map_tasks = context.getCounter(MyCounter.MAP_TASKS);
Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
map_tasks.increment(1);
total_tasks.increment(1);
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.set(new Object[] { 1L });
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
for (int i = 0;
i < record.getColumnCount();
i++) {
word.set(new Object[] { record.get(i).toString() });
context.write(word, one);
}
}
}
public static class SumReducer extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException { result = context.createOutputRecord();
Counter reduce_tasks = context.getCounter(MyCounter.REDUCE_TASKS);
Counter total_tasks = context.getCounter(MyCounter.TOTAL_TASKS);
reduce_tasks.increment(1);
total_tasks.increment(1);
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
long count = 0;
while (values.hasNext()) {
Record val = values.next();
```

```
count += (Long) val.get(0);
}
result.set(0, key.get(0));
result.set(1, count);
context.write(result);
}
}
public static void main(String[] args) throws Exception { if (args.length ! = 2) {
System.err.println("Usage: TestUserDefinedCounters <in_table> <out_table>");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(TokenizerMapper.class);
job.setReducerClass(SumReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
RunningJob rJob = JobClient.runJob(job);
Counters counters = rJob.getCounters();
long m = counters.findCounter(MyCounter.MAP_TASKS).getValue();
long r = counters.findCounter(MyCounter.REDUCE_TASKS).getValue();
long total = counters.findCounter(MyCounter.TOTAL_TASKS).getValue();
System.exit(0);
}
}
```

1.8.6.8. grep example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.Mapper;
import com.aliyun.odps.mapred.MapperBase;
```

```
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* Extracts matching regexs from input files and counts them.
*
* To run: jar -resources odps-mapred-example-0.12.0.jar
* com.aliyun.odps.mapred.open.example.Grep mr_src mr_grep_tmp > mr_grep_out val;
**/
public class Grep {
/**
* RegexMapper
**/
public class RegexMapper extends MapperBase { private Pattern pattern;
private int group;
private Record word;
private Record one;
@Override
public void setup(TaskContext context) throws IOException {
JobConf job = (JobConf) context.getJobConf();
pattern = Pattern.compile(job.get("mapred.mapper.regex"));
group = job.getInt("mapred.mapper.regex.group", 0);
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.set(new Object[] { 1L });
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
for (int i = 0; i < record.getColumnCount(); ++i) {
String text = record.get(i).toString();
Matcher = pattern. matcher (text );
while (matcher.find()) {
word.set(new Object[] { matcher.group(group) });
context.write(word, one);
}
}
}
```

```
}
}
/**
* LongSumReducer
**/
public class LongSumReducer extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException { result = context.createOutputRecord();
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
Long Count = 0;
while (values.hasNext()) {
Record val = values.next();
count += (Long) val.get(0);
}
result.set(0, key.get(0));
result.set(1, count);
context.write(result);
}
}
/**
* A {@link Mapper} that swaps keys and values.
**/
public class InverseMapper extends MapperBase {
private Record word;
private Record count;
@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputValueRecord();
count = context.createMapOutputKeyRecord();
}
/**
* The inverse function. Input keys and values are swapped.
**/
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
word.set(new Object[] { record.get(0).toString() });
count.set(new Object[] { (Long) record.get(1) });
context.write(count, word);
```

```
}
}
/**
* IdentityReducer
**/
public class IdentityReducer extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException {
result = context.createOutputRecord();
}
/** Writes all keys and values directly to output. **/
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
result.set(0, key.get(0));
while (values.hasNext()) {
Record val = values.next();
result.set(1, val.get(0));
context.write(result);
}
}
}
public static void main(String[] args) throws Exception {
if (args.length < 4) {
System.err.println("Grep <inDir> <tmpDir> <outDir> <regex> [<group>]"); System.exit(2);
}
JobConf grepJob = new JobConf();
grepJob.setMapperClass(RegexMapper.class);
grepJob.setReducerClass(LongSumReducer.class);
grepJob.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
grepJob.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), grepJob);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), grepJob);
grepJob.set("mapred.mapper.regex", args[3]);
if (args.length == 5) {
grepJob.set("mapred.mapper.regex.group", args[4]);
}
@SuppressWarnings("unused")
RunningJob rjGrep = JobClient.runJob(grepJob);
JobConf sortJob = new JobConf();
sortJob.setMapperClass(InverseMapper.class);
```

```
sortJob.setReducerClass(IdentityReducer.class);
sortJob.setMapOutputKeySchema(SchemaUtils.fromString("count:bigint"));
sortJob.setMapOutputValueSchema(SchemaUtils.fromString("word:string"));-
InputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), sortJob);
OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), sortJob);
sortJob.setNumReduceTasks(1); // write a single file
sortJob.setOutputKeySortColumns(new String[] { "count" });
 // sort by
// decreasing
// freq
@SuppressWarnings("unused")
RunningJob rjSort = JobClient.runJob(sortJob);
}
}
```

1.8.6.9. JOIN example

The MaxCompute MapReduce framework does not support the JOIN logic. However, you can use custom Map and Reduce functions to complete JOIN operations. This requires extra work.

Table mr_join_src1(key bigint, value string) must be joined with mr_join_src2(key bigint, value string). The output table is mr_join_out(key bigint, value1 string, value2 string), where value1 indicates the value of mr_join_src1 and value2 indicates the value of mr_join_src2.

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
Import java. util. arraylist;
import java.util.Iterator;
import java.util.List;
import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
```

```
* Join
*
* To run: jar -resources odps-mapred-example-0.12.0.jar
* com.aliyun.odps.mapred.open.example.Join mr_join_src11 > mr_join_src22 mr_join_out;
*
**/
public class Join {
public static class JoinMapper extends MapperBase {
private Record mapkey;
private Record mapvalue;
@Override
public void setup(TaskContext context) throws IOException {
mapkey = context.createMapOutputKeyRecord();
mapvalue = context.createMapOutputValueRecord();
}
@Override
public void map(long key, Record record, TaskContext context) throws IOException {
/* Determine the source table of the record based on the value field. This is the user code logic. If the
source table of the record cannot be determined based on the value field, you can add a field in the in
put table. The tag generated based on the value field is used in connection operations in the Reduce s
tage. */
long tag = 1;
String val = record.get(1).toString();
if (val.startsWith("valb_")) {
tag = 2;
}
mapkey.set(0, Long.parseLong(record.get(0).toString()));
mapkey.set(1, tag);
mapvalue.set(0, tag);
for (int i = 1; i < record.getColumnCount(); i++) {
mapvalue.set(i, record.get(i));
}
context.write(mapkey, mapvalue);
}
}
public static class JoinReducer extends ReducerBase {
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException {
result = context.createOutputRecord();
}
```

```
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
long k = (Long) key.get(0);
List<Object[]> list1 = new ArrayList<Object[]>();
Counter cnt = context.getCounter("MyCounters", "reduce_outputs");
 cnt.increment(1);
while (values.hasNext()) {
Record value = values.next();
long tag = (Long) value.get(0);
if (tag == 1) {
//If the data comes from the first table, the data is cached in the list.
//Data is sorted by key and tag, so the value with tag==1 is sorted on the top.
//We recommended that you exercise caution when using this method in practice. If a key has many va
lues,
//the memory usage of Reduce stage increases. When the memory usage exceeds the value set by Job
Conf::setMemoryForReduceTask,
//the Reduce stage may be terminated by the system due to memory overflow.
list1.add(value.toArray().clone());
} else {
//If the data comes from the second table, the data is sorted by key and tag.
//All values in the first table have been saved in list1.
//For the values in the first table
for (Object r1: list1) { int index = 0;
//Set the key first.
result.set(index++, k);
Object[] s_arr = (Object[])r1;
result.set(index++, s_arr[1].toString());
result.set(index++, value.get(1).toString());
context.write(result);
}
}
}
}
}
public static void main(String[] args) throws Exception {
if (args.length ! = 3) {
System.err.println("Usage: Join <input table1> <input table2> <out>");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(JoinMapper.class);
```

```
job.setReducerClass(JoinReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,tag:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("tagx:bigint,value:string"));
job.setPartitionColumns(new String[] { "key" });
//Sort data by key and tag. The JOIN operation can be performed in the Reduce stage only after data s
orting by key.
//The tag indicates the source table of the current record.
job.setOutputKeySortColumns(new String[] { "key", "tag" });
job.setOutputGroupingColumns(new String[] { "key" });
//The Reduce stage uses lists to cache data. Therefore, we recommend that you increase the memory
size for Reduce workers.
job.setMemoryForReduceTask(4096);
job.setInt("table.counter", 0);
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
InputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[2]).build(), job);
JobClient.runJob(job);
}
}
```

1.8.6.10. Sleep example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
Import java. util. iterator;
importjava.util.LinkedHashMap;
import java.util.Map;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import com.aliyun.odps.OdpsException;
Import com. aliyun. ODPS. Data. record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
```

```
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* Dummy class for testing MR framefork. Sleeps for a defined period of > time in
* mapper and reducer. Generates fake input for map / reduce jobs. Note > that
* generated number of input pairs is in the order of
* <code>numMappers
* mapSleepTime / 100</code>, so the > job uses some disk
* space.
* To run: jar -resources odps-mapred-example-0.12.0.jar > com.aliyun.odps.mapred.open.example.Slee
pJob
* -m 1 -r 1 -mt 1 -rt 1;
*
**/
public class SleepJob {
private static Log LOG = LogFactory.getLog(SleepJob.class);
public static class SleepMapper extends MapperBase {
private LinkedHashMap<Integer, Integer> inputs = new LinkedHashMap<Integer, Integer>();
private long mapSleepDuration = 100;
private int mapSleepCount = 1;
private int count = 0;
private Record key;
@Override
public void setup(TaskContext context) throws IOException{
LOG.info("map setup called");
JobConf conf = (JobConf) context.getJobConf();
mapSleepCount = conf.getInt("sleep.job.map.sleep.count", 1);
if (mapSleepCount < 0)
throw new IOException("Invalid map count: " + mapSleepCount);
mapSleepDuration = conf.getLong("sleep.job.map.sleep.time", 100)
/ mapSleepCount;
LOG.info("mapSleepCount = " + mapSleepCount + ", mapSleepDuration = " + mapSleepDuration);
final int redcount = conf.getInt("sleep.job.reduce.sleep.count", 1);
if (redcount < 0)
throw new IOException("Invalid reduce count: " + redcount);
final int emitPerMapTask = (redcount * conf.getNumReduceTasks());
int records = 0;
int emitCount = 0;
while (records++ < mapSleepCount) {
int key = emitCount;
int emit = emitPerMapTask / mapSleepCount;
if ((emitPerMapTask) % mapSleepCount > records) {
```

```
if ((emit+1+mapTask) % mapSleepCount > records) {
++emit;
}
emitCount += emit;
int value = emit;
inputs.put(key, value);
}
key = context.createMapOutputKeyRecord();
}
@Override
public void cleanup(TaskContext context) throws IOException {
// it is expected that every map processes mapSleepCount number of
// records.
LOG.info("map run called");
for (Map.Entry<Integer, Integer> entry : inputs.entrySet()) {
LOG.info("Sleeping... (" + (mapSleepDuration * (mapSleepCount - count)) + ") ms left");
try {
Thread.sleep(mapSleepDuration);
} catch (InterruptedException e) { throw new IOException(e);
}
++count;
// output reduceSleepCount * numReduce number of random values, so that
// each reducer will get reduceSleepCount number of keys.
int k = entry.getKey();
int v = entry.getValue();
for (int i = 0; i < v; ++i) {
key.set(new Object[] { (Long) ((long) (k + i)) });
context.write(key, key);
}
}
}
}
public static class SleepReducer extends ReducerBase {
private long reduceSleepDuration = 100;
private int reduceSleepCount = 1;
private int count = 0;
@Override
public void setup(TaskContext context) throws IOException {
LOG.info("reduce setup called");
JobConf conf = (JobConf) context.getJobConf();
reduceSleepCount = conf.getInt("sleep.job.reduce.sleep.count",
```

```
reduceSleepCount);
reduceSleepDuration = conf.getLong("sleep.job.reduce.sleep.time", 100)
/ reduceSleepCount;
LOG.info("reduceSleepCount = " + reduceSleepCount
+ ", reduceSleepDuration = " + reduceSleepDuration);
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
LOG.info("reduce called");
LOG.info("Sleeping... ("
+ (reduceSleepDuration * (reduceSleepCount - count)) + ") ms left"); try {
Thread.sleep(reduceSleepDuration);
} catch (InterruptedException e) {
throw new IOException(e);
}
count++;
}
}
public static int run(int numMapper, int numReducer, long mapSleepTime, int mapSleepCount, long red
uceSleepTime, int reduceSleepCount)
throws OdpsException {
JobConf job = setupJobConf(numMapper, numReducer, mapSleepTime, mapSleepCount, reduceSleepTi
me, reduceSleepCount);
JobClient.runJob(job);
return 0;
}
public static JobConf setupJobConf(int numMapper, int numReducer, long mapSleepTime, int mapSleep
Count, long reduceSleepTime, int reduceSleepCount) {
JobConf job = new JobConf();
InputUtils.addTable(TableInfo.builder().tableName("mr_empty").build(), job);
OutputUtils.addTable(TableInfo.builder().tableName("mr_sleep_out").build(), job);
job.setNumReduceTasks(numReducer);
job.setMapperClass(SleepMapper.class);
job.setReducerClass(SleepReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("int1:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("int2:bigint"));
job.setPartitionColumns(new String[] { "int1" });
job.setLong("sleep.job.map.sleep.time", mapSleepTime);
job.setLong("sleep.job.reduce.sleep.time", reduceSleepTime);
job.setInt("sleep.job.map.sleep.count", mapSleepCount);
job.setInt("sleep.job.reduce.sleep.count", reduceSleepCount);
```

```
return job;
}
private static void printUsage() {
System.err.println("SleepJob [-m numMapper] [-r numReducer]"
+ " [-mt mapSleepTime (msec)] [-rt reduceSleepTime (msec)]"
+ " [-recordt recordSleepTime (msec)]");
}
public static void main(String[] args) throws Exception {
if (args.length < 1) {
printUsage();
return;
}
int numMapper = 1, numReducer = 1;
long mapSleepTime = 100, reduceSleepTime = 100, recSleepTime = 100;
int mapSleepCount = 1, reduceSleepCount = 1;
for (int i = 0; i < args.length; i++) { if (args[i].equals("-m")) {
numMapper = Integer.parseInt(args[++i]);
} else if (args[i].equals("-r")) {
numReducer = Integer.parseInt(args[++i]);
} else if (args[i].equals("-mt")) {
mapSleepTime = Long.parseLong(args[++i]);
} else if (args[i].equals("-rt")) {
reduceSleepTime = Long.parseLong(args[++i]);
} else if (args[i].equals("-recordt")) { recSleepTime = Long.parseLong(args[++i]);
}
}
// sleep for *SleepTime duration in Task by recSleepTime per record
mapSleepCount = (int) Math.ceil(mapSleepTime / ((double) recSleepTime));
reduceSleepCount = (int) Math.ceil(reduceSleepTime
/ ((double) recSleepTime));
run(numMapper, numReducer, mapSleepTime, mapSleepCount, reduceSleepTime, reduceSleepCount);
}
}
```

1.8.6.11. unique example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException;
import java.util.Iterator;
```

```
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* Unique Remove duplicate words
*
* To run: jar -resources odps-mapred-example-0.12.0.jar
* com.aliyun.odps.open.mapred.example.Unique mr_sort_in > mr_unique_out
* key|value|all;
*
**/
public class Unique {
public static class OutputSchemaMapper extends MapperBase {
private Record key;
private Record value;
@Override
public void setup(TaskContext context) throws IOException {
key = context.createMapOutputKeyRecord();
value = context.createMapOutputValueRecord();
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
long left = 0;
long right = 0;
if (record.getColumnCount() > 0) { left = (Long) record.get(0);
if (record.getColumnCount() > 1) { right = (Long) record.get(1);
}
key.set(new Object[] { (Long) left, (Long) right });
value.set(new Object[] { (Long) left, (Long) right });
context.write(key, value);
}
}
}
public static class OutputSchemaReducer extends ReducerBase {
```

```
private Record result = null;
@Override
public void setup(TaskContext context) throws IOException {
result = context.createOutputRecord();
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
result.set(0, key.get(0));
while (values.hasNext()) {
Record value = values.next();
result.set(1, value.get(1));
}
context.write(result);
}
}
public static void main(String[] args) throws Exception {
if (args.length > 3 || args.length < 2) {
System.err.println("Usage: unique <in> <out> [key|value|all]");
System.exit(2);
}
String ops = "all";
if (args.length == 3) { ops = args[2];
}
// Key Unique
if (ops.equals("key")) {
JobConf job = new JobConf();
job.setMapperClass(OutputSchemaMapper.class);
job.setReducerClass(OutputSchemaReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setPartitionColumns(new String[] { "key" });
job.setOutputKeySortColumns(new String[] { "key", "value" });
job.setOutputGroupingColumns(new String[] { "key" });
job.set("tablename2", args[1]); job.setNumReduceTasks(1);
job.setInt("table.counter", 0);
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
// Key&Value Unique
```

```
if (ops.equals("all")) {
JobConf job = new JobConf();
job.setMapperClass(OutputSchemaMapper.class);
job.setReducerClass(OutputSchemaReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setPartitionColumns(new String[] { "key" });
job.setOutputKeySortColumns(new String[] { "key", "value" });
job.setOutputGroupingColumns(new String[] { "key", "value" });
job.set("tablename2", args[1]); job.setNumReduceTasks(1);
job.setInt("table.counter", 0);
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
// Value Unique
if (ops.equals("value")) { JobConf job = new JobConf();
job.setMapperClass(OutputSchemaMapper.class);
job.setReducerClass(OutputSchemaReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setMapOutputValueSchema(SchemaUtils.fromString("key:bigint,value:bigint"));
job.setPartitionColumns(new String[] { "value" });
job.setOutputKeySortColumns(new String[] { "value" });
job.setOutputGroupingColumns(new String[] { "value" });
job.set("tablename2", args[1]); job.setNumReduceTasks(1);
job.setInt("table.counter", 0);
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), job);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
}
}
```

1.8.6.12. Sort example

Example:

```
package com.aliyun.odps.mapred.open.example;
import java.io.IOException; import java.util.Date;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
 import com.aliyun.odps.mapred.JobClient;
```

```
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.example.lib.IdentityReducer;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
/**
* This is the trivial map/reduce program that does absolutely nothing > other
* than use the framework to fragment and sort the input values.
*
* To run: jar -resources odps-mapred-example-0.12.0.jar > com.aliyun.odps.mapred.open.example.Sort
* mr_sort_in mr_sort_out;
*
**/
public class Sort {
static int printUsage() {
System.out.println("sort <input> <output>"); return -1;
}
/**
* Implement the identity function, mapping record's first two columns > to
* outputs.
**/
public static class IdentityMapper extends MapperBase {
private Record key;
private Record value;
@Override
public void setup(TaskContext context) throws IOException {
key = context.createMapOutputKeyRecord();
value = context.createMapOutputValueRecord();
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
Key.set (new object [] {(long) record.get (0 )});
value.set(new Object[] { (Long) record.get(1) });
context.write(key, value);
}
}
/**
* The main driver for sort program. Invoke this method to submit the
```

```
* map/reduce job.
*
* @throws IOException
* When there is communication problems with the job tracker.
**/
public static void main(String[] args) throws Exception {
JobConf jobConf = new JobConf();
jobConf.setMapperClass(IdentityMapper.class);
jobConf.setReducerClass(IdentityReducer.class);
jobConf.setNumReduceTasks(1);
Jobconf.setmapoutputkeyschema schemautils schemeiutils.fromstring ("key: bigint "));
jobConf.setMapOutputValueSchema(SchemaUtils.fromString("value:bigint"));
InputUtils.addTable(TableInfo.builder().tableName(args[0]).build(), jobConf);
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), jobConf);
Date startTime = new Date(); System.out.println("Job started: " + startTime);
JobClient.runJob(jobConf);
Date end_time = new Date(); System.out.println("Job ended: " + end_time); System.out.println("The jo
b took " + (end_time.getTime() - startTime.getTime()) / 1000 + " seconds.") ;
}
}
```

1.8.6.13. Example of using partitioned table as an input

The following examples use partitions as an input.

Example 1:

```
public static void main(String[] args) throws Exception {
JobConf job = new JobConf();
...
LinkedHashMap<String, String> input = new LinkedHashMap<String, String>();
input.put("pt", "123456");
InputUtils.addTable(TableInfo.builder().tableName("input_table").partSpec(input).build(), job);
LinkedHashMap<String, String> output = new LinkedHashMap<String, String>(); output.put("ds", "6543
21");
OutputUtils.addTable(TableInfo.builder().tableName("output_table").partSpec(output).build(), job);
JobClient.runJob(job);
}
```

Example 2:

```
package com.aliyun.odps.mapred.open.example;
...
public static void main(String[] args) throws Exception {
if (args.length ! = 2) {
System.err.println("Usage: WordCount <in_table> <out_table>");
System.exit(2);
}
JobConf job = new JobConf();
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(SumCombiner.class);
job.setReducerClass(SumReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
Account account = new AliyunAccount("my_access_id", "my_access_key");
Odps odps = new Odps(account);
odps.setEndpoint("odps_endpoint_url");
odps.setDefaultProject("my_project");
Table table = odps.tables().get(tblname);
TableInfoBuilder builder = TableInfo.builder().tableName(tblname);
for (Partition p : table.getPartitions()) { if (applicable(p)) {
LinkedHashMap<String, String> partSpec = new LinkedHashMap<String, String>();
for (String key : p.getPartitionSpec().keys()) {
partSpec.put(key, p.getPartitionSpec().get(key));
}
InputUtils.addTable(builder.partSpec(partSpec).build(), conf);
}
}
OutputUtils.addTable(TableInfo.builder().tableName(args[1]).build(), job);
JobClient.runJob(job);
}
```

Note In the preceding example, the MaxCompute SDK and MapReduce SDK are used together to allow MapReduce tasks to read data from partitions. The preceding code cannot be compiled for execution. It is just an example of the main function. In the preceding example, the applicable function is the user logic that determines whether the partitions can be used as an input of the MapReduce job.

1.8.6.14. Pipeline example

Example:

```
package com.aliyun.odps.mapred.example;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.Column;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.OdpsType;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.Job;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.pipeline.Pipeline;
public class Histogram {
public static class TokenizerMapper extends MapperBase {
Record word;
Record one;
@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.setBigint(0, 1L);
}
@Override
public void map(long recordNum, Record record, TaskContext context)
throws IOException {
for (int i = 0; i < record.getColumnCount(); i++) {
String[] words = record.get(i).toString().split("\\s+");
for (String w : words) {
word.setString(0, w);
context.write(word, one);
}
}
}
}
public static class SumReducer extends ReducerBase { private Record num;
private Record result;
@Override
public void setup(TaskContext context) throws IOException {
num = context.createOutputKeyRecord();
result = context.createOutputValueRecord();
```

```
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
long count = 0;
while (values.hasNext()) {
Record val = values.next();
count += (Long) val.get(0);
}
result.set(0, key.get(0));
num.set(0, count);
context.write(num, result);
}
}
public static class IdentityReducer extends ReducerBase {
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException {
while (values.hasNext()) {
context.write(values.next());
}
}
}
public static void main(String[] args) throws OdpsException {
if (args.length ! = 2) {
System.err.println("Usage: orderedwordcount <in_table> <out_table>");
System.exit(2);
}
Job job = new Job();
/***
* In the process of constructing pipeline, if you do not specify mapper's OutputKeySortColumns, Partiti
onColumns, OutputGroupingColumns,
* the framework defaults to its OutputKey as the default configuration for the three
***/
Pipeline pipeline = Pipeline.builder()
.addMapper(TokenizerMapper.class)
.setOutputKeySchema(
new Column[] { new Column("word", OdpsType.STRING) })
.setOutputValueSchema(
new Column[] { new Column("count", OdpsType.BIGINT) })
.setOutputKeySortColumns(new String[] { "word" })
.setPartitionColumns(new String[] { "word" })
.setOutputGroupingColumns(new String[] { "word" })
```

```
.addReducer(SumReducer.class)
.setOutputKeySchema(
new Column[] { new Column("count", OdpsType.BIGINT) })
.setOutputValueSchema(
new Column[] { new Column("word", OdpsType.STRING)})
.addReducer(IdentityReducer.class).createPipeline();
job. setpipeline (pipeline);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
job.submit(); job.waitForCompletion();
System.exit(job.isSuccessful() == true ? 0 : 1);
}
}
```

1.9. MaxCompute Graph

1.9.1. Graph overview

1.9.1.1. Graph overview

MaxCompute Graph is a processing framework designed for iterative graph computing. Graph computing jobs use graphs to build models. Graphs are composed of vertices and edges with values. MaxCompute Graph supports the following operations to edit a graph:

- Editing the value of vertex or edge.
- Adding/deleting vertex.
- Adding/deleting edge.

> **Note** When editing the vertex or edge, you must maintain the relationship between the two items.

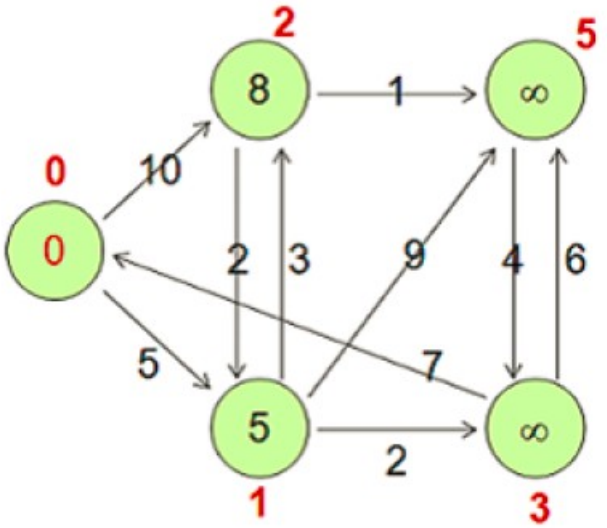
After performing iterative graph editing and evolution, you can get the final result. Typical applications include PageRank, SSSP algorithm, and K-Means algorithm. Furthermore, you can use the Java SDK provided by MaxCompute Graph to compile computing programs.

1.9.1.2. Graph data structure

MaxCompute Graph processes directed graphs, or digraphs, that consist of a vertex and an edge. MaxCompute stores data in two-dimensional tables, so you must convert graph data into two-dimensional tables and store them in MaxCompute. To perform graph analysis, you must use a custom GraphLoader to convert two-dimensional table data to vertexes and edges in the MaxCompute Graph engine. You can then determine how to break down and analyze your graph data based on your business requirements. In the following chapter, the examples provided use different tabular expressions to represent the data structure of a graph.

The vertex structure can be expressed as < ID, Value, Halted, Edges >, indicating respectively the vertex identifier, the value, the state (Halted, meaning whether to stop iteration), and the edge set (Edges, indicating lists of all edges starting from the vertex). The edge structure can be described as < DestVertexID, Value >, indicating respectively the destination vertex (DestVertexID) and value (Value). The following figure shows Graph data structure.

Graph data structure



The preceding figure involves the following vertexes.

Graph data structure

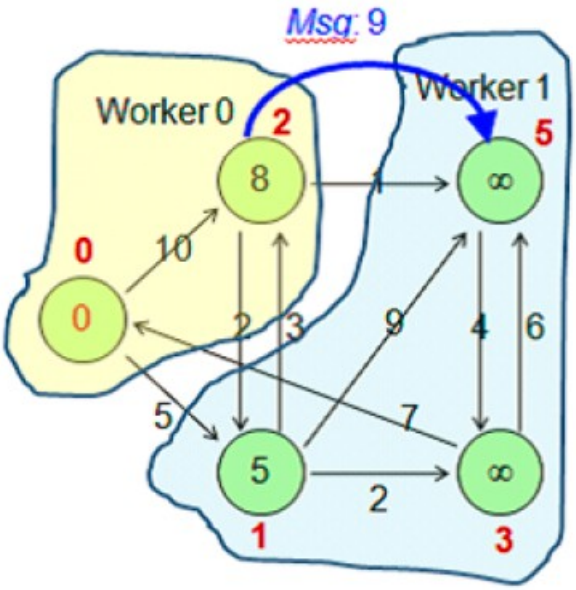
Vertex	<ID, Value, Halted, Edges>
v0	<0, 0, false, [<1, 5 >, <2, 10 >] >
v1	<1, 5, false, [<2, 3>, <3, 2>, <5, 9>]>
v2	<2, 8, false, [<1, 2>, <5, 1 >]>
v3	<3, Long.MAX_VALUE, false, [<0, 7>, <5, 6>]>
v5	<5, Long.MAX_VALUE, false, [<3, 4 >]>

1.9.1.3. Graph logic

1.9.1.3.1. Load graph

- Graph load: The framework calls the custom GraphLoader to parse the records in the input table into vertices or edges.
- Partitioning: The framework calls the custom Partitioner to partition vertices (default partition logic: the hash value of the vertex ID modulo the number of workers). Then, the framework distributes the partitions to the corresponding workers.

Load graph



Based on the preceding figure, if there are two workers, v0 and v2 are distributed to Worker0, because the result of the ID modulo 2 (total number of workers) is 0. v1, v3, and v5 are distributed to Worker1, because the result of the ID modulo 2 is 1.

1.9.1.3.2. Iterative computation

An iteration is a superstep. During a superstep, all vertices not in the halted state (Halted value is false) or vertices that receive messages (vertices in the halted state automatically wake up when receiving a message) are traversed. The compute method (ComputeContext context, Iterable messages) of these vertices is called.

In a custom compute method (ComputeContext context, Iterable messages):

- The messages sent by the previous superstep to the current vertex are processed.
- The graph is edited as required:
  - The values of vertices or edges are modified.
  - Messages are sent to some vertices.
  - Vertices or edges are added or deleted.
- The aggregator aggregates information to obtain global information.
- The current vertex is set to the halted or non-halted state.
- During each iteration, the framework automatically sends messages to the corresponding workers asynchronously. The messages are processed in the next superstep.

1.9.1.3.3. End of iteration

An iteration ends if any of the following conditions is satisfied.

- All vertices are in the halted state (Halted value is true), and no new messages are generated.
- The maximum number of iterations is reached.
- The terminate method of an aggregator returns true.

Example:

```
// 1. Load
for each record in input_table { GraphLoader.load();
}
// 2. Setup
WorkerComputer.setup();
for each aggr in aggregators { aggr.createStartupValue();
}
for each v in vertices { v.setup();
}
// 3. Superstep
for (step = 0; step < max; step ++) { for each aggr in aggregators { aggr.createInitialValue();
}
for each v in vertices { v.compute();
}
}
// 4. Cleanup
for each v in vertices { v.cleanup();
}
WorkerComputer.cleanup();
```

1.9.1.4. Aggregator overview

Aggregator is a common feature in MaxCompute Graph jobs and is especially suited for solving machine learning issues. In MaxCompute Graph, Aggregator is used to summarize and process global information.

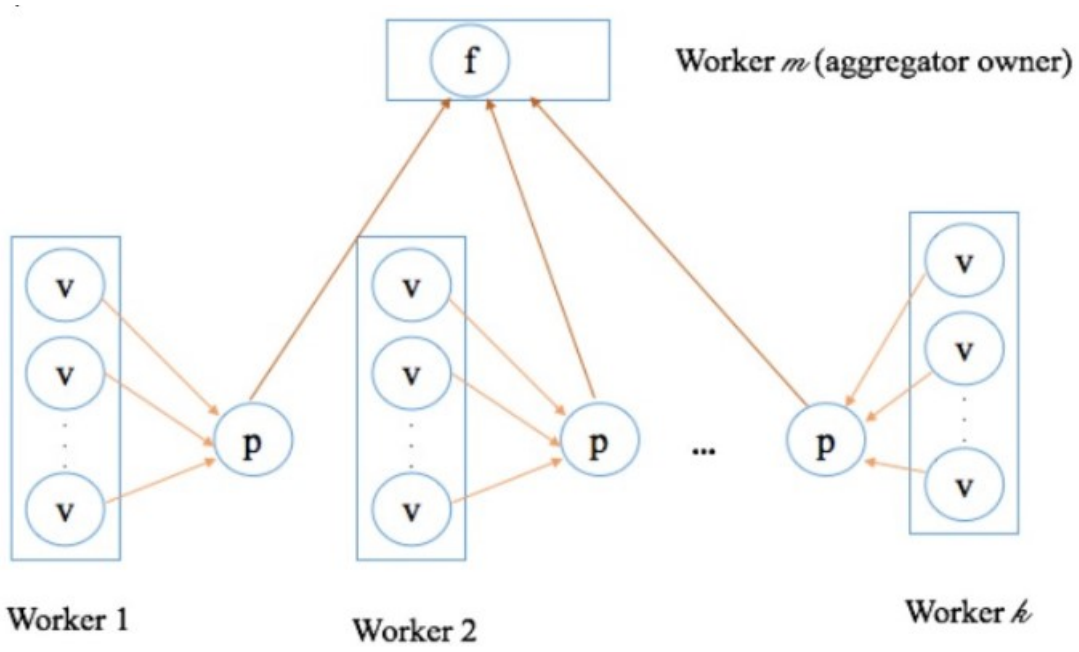
This topic describes the implementation mechanism and related API operations of Aggregator and uses Kmeans Clustering as an example to demonstrate how to use Aggregator.

Implementation mechanism

The logic of Aggregator is divided into two parts:

- One part is implemented on all Workers in distributed mode.
- The other part is only implemented on the Worker where Aggregator Owner is located in single vertex mode.

Initial values are created and partially aggregated on each Worker, and then the partial aggregation results of all Workers are sent to the Worker where Aggregator Owner is located. This Worker then aggregates the received partial aggregation objects into a global aggregation result and determines whether to end the iteration. The global aggregation result is distributed to all Workers in the next superstep for iteration.



API operations

Aggregator provides five API operations. The following parts describe when to call these API operations and for what purposes.

- **createStartupValue(context)**

  This API operation is performed once on all Workers before all supersteps start. It is used to initialize AggregatorValue. In the first superstep iteration (superstep 0), WorkerContext.getLastAggregatedValue() or ComputeContext.getLastAggregatedValue() is called to obtain the initialized AggregatorValue object.

- **createInitialValue(context)**

  This API operation is performed once on all Workers when each superstep starts. It is used to initialize AggregatorValue for the current iteration. Generally, WorkerContext.getLastAggregatedValue() is called to obtain the result of the previous iteration, and then partial initialization is implemented.

- **aggregate(value, item)**

  This API operation is also performed on all Workers. It is triggered by an explicit call to ComputeContext#aggregate(item), while the preceding two API operations are automatically called by the framework.

This API operation is used to implement partial aggregation. The first parameter value indicates the aggregation result of the Worker in the current superstep. The initial value is the object returned by createInitialValue. The second parameter is passed in when ComputeContext#aggregate(item) is called by using user code. In this API operation, item is typically used to update value for aggregation. After all the aggregate operations are performed, the obtained value is the partial aggregation result of the Worker. The result is then sent by the framework to the Worker where Aggregator Owner is located.

- **merge(value, partial)**

  This API operation is performed on the Worker where Aggregator Owner is located. It is used to merge partial aggregation results of Workers to obtain the global aggregation object. Similar to aggregate, value in this API operation indicates the aggregated results, while partial indicates objects to be aggregated. partial is used to update value.

  Assume that there are three Workers w0, w1, and w2, and the corresponding partial aggregation results are p0, p1, and p2. If p1, p0, and p2 are sent to the Worker where Aggregator Owner is located in sequence, the following merge operations are performed:

  i. merge(p1, p0) is executed first to aggregate p1 and p0 as p1.
  ii. Then, merge(p1, p2) is executed to aggregate p1 and p2 as p1. p1 is the global aggregation result in this superstep.

  Therefore, when only one Worker exists, the merge method is not required. In this case, merge() is not called.

- **terminate(context, value)**

  After the Worker where Aggregator Owner is located executes merge(), the framework calls terminate(context, value) to perform the final processing. The second parameter value indicates the global aggregation result obtained by calling merge(). The global aggregation result can be further modified in this API operation. After terminate() is executed, the framework distributes the global aggregation object to all Workers for the next superstep.

  If true is returned for terminate(), iteration is ended for the entire job. Otherwise, the iteration continues. In machine learning scenarios, jobs end generally when true is returned after convergence.

Kmeans Clustering example

This section uses typical Kmeans Clustering as an example to demonstrate how to use Aggregator.

> **Note** If you need the complete code, see Kmeans. In this section, the code is resolved and is for reference only.

- GraphLoader

  GraphLoader is used to load an input table and convert it to vertices or edges of a graph. In this example, each row of data in the input table is a sample, each sample constructs a vertex, and vertex values are used to store samples.

  A Writable class KmeansValue is defined as the value type of a vertex.

```
public static class KmeansValue implements Writable {
   DenseVector sample;
   public KmeansValue() {
   }
   public KmeansValue(DenseVector v) {
      this.sample = v;
   }
   @Override
      public void write(DataOutput out) throws IOException {
      wirteForDenseVector(out, sample);
   }
   @Override
      public void readFields(DataInput in) throws IOException {
      sample = readFieldsForDenseVector(in);
   }
}
```

A DenseVector object is encapsulated in KmeansValue to store a sample. The DenseVector type stems from matrix-toolkits-java. wirteForDenseVector() and readFieldsForDenseVector() are used for serialization and deserialization. For more information, see the complete code.

Custom KmeansReader code:

```
public static class KmeansReader extends
   GraphLoader<LongWritable, KmeansValue, NullWritable, NullWritable> {
   @Override
      public void load(
      LongWritable recordNum,
      WritableRecord record,
      MutationContext<LongWritable, KmeansValue, NullWritable, NullWritable> context)
      throws IOException {
      KmeansVertex v = new KmeansVertex();
      v.setId(recordNum);
      int n = record.size();
      DenseVector dv = new DenseVector(n);
      for (int i = 0; i < n; i++) {
         dv.set(i, ((DoubleWritable)record.get(i)).get());
      }
      v.setValue(new KmeansValue(dv));
      context.addVertexRequest(v);
   }
}
```

In KmeansReader, a vertex is created when each row of data (a record) is read. recordNum is used as the vertex ID, and the record content is converted to a DenseVector object and encapsulated in VertexValue.

- Vertex

  Custom KmeansVertex code:

```
public static class KmeansVertex extends
   Vertex<LongWritable, KmeansValue, NullWritable, NullWritable> {
   @Override
      public void compute(
      ComputeContext<LongWritable, KmeansValue, NullWritable, NullWritable> context,
      Iterable<NullWritable> messages) throws IOException {
      context.aggregate(getValue());
   }
}
```

  The logic of the preceding code is to implement partial aggregation for samples maintained for each iteration. For more information about the logic, see the implementation of Aggregator in the following section.

- Aggregator

  The main logic of Kmeans is concentrated on Aggregator. Custom KmeansAggrValue is used to maintain the content you want to aggregate and distribute.

```
public static class KmeansAggrValue implements Writable {
   DenseMatrix centroids;
   DenseMatrix sums; // used to recalculate new centroids
   DenseVector counts; // used to recalculate new centroids
   @Override
      public void write(DataOutput out) throws IOException {
      wirteForDenseDenseMatrix(out, centroids);
      wirteForDenseDenseMatrix(out, sums);
      wirteForDenseVector(out, counts);
   }
   @Override
      public void readFields(DataInput in) throws IOException {
      centroids = readFieldsForDenseMatrix(in);
      sums = readFieldsForDenseMatrix(in);
      counts = readFieldsForDenseVector(in);
   }
}
```

  In the preceding code, three objects are maintained in KmeansAggrValue:

- **centroids**: indicates the existing K centers. If the sample is m-dimensional, centroids is a matrix of K × m.
- **sums**: indicates a matrix of the same size as centroids. Each element records the sum of a specific dimension of the sample closest to a specific center. For example, sums(i,j) indicates the sum of dimension j of the sample closest to center i.
- **counts** is a K-dimensional vector. It records the number of samples closest to each center. counts is used with sums to calculate a new center, which is the main content to be aggregated.

KmeansAggregator is a custom Aggregator implementation class. The implementation is described below in order of the preceding API operations:

i. Implementation of **createStartupValue()**

```
public static class KmeansAggregator extends Aggregator<KmeansAggrValue> {
   public KmeansAggrValue createStartupValue(WorkerContext context) throws IOException {
      KmeansAggrValue av = new KmeansAggrValue();
      byte[] centers = context.readCacheFile("centers");
      String lines[] = new String(centers).split("\n");
      int rows = lines.length;
      int cols = lines[0].split(",").length; // assumption rows >= 1
      av.centroids = new DenseMatrix(rows, cols);
      av.sums = new DenseMatrix(rows, cols);
      av.sums.zero();
      av.counts = new DenseVector(rows);
      av.counts.zero();
      for (int i = 0; i < lines.length; i++) {
         String[] ss = lines[i].split(",");
         for (int j = 0; j < ss.length; j++) {
            av.centroids.set(i, j, Double.valueOf(ss[j]));
         }
      }
      return av;
   }
```

This method initializes a KmeansAggrValue object, reads the initial center from the resource file centers, and assigns a value to centroids. The initial values of sums and counts are 0.

ii. Implementation of **createInitialValue()**

```
@Override
public KmeansAggrValue createInitialValue(WorkerContext context)
    throws IOException {
    KmeansAggrValue av = (KmeansAggrValue)context.getLastAggregatedValue(0);
    // reset for next iteration
    av.sums.zero();
    av.counts.zero();
    return av;
}
```

This method first obtains KmeansAggrValue of the previous iteration, and then clears the values of sums and counts. Only the centroids value of the previous iteration is retained.

iii. Implementation of **aggregate()**

```
@Override
public void aggregate(KmeansAggrValue value, Object item)
    throws IOException {
    DenseVector sample = ((KmeansValue)item).sample;
    // find the nearest centroid
    int min = findNearestCentroid(value.centroids, sample);
    // update sum and count
    for (int i = 0; i < sample.size(); i ++) {
        value.sums.add(min, i, sample.get(i));
    }
    value.counts.add(min, 1.0d);
}
```

This method calls findNearestCentroid() to find the index of the center closest to the sample item, uses sums to add up all dimensions, and increments the value of counts by 1.

The preceding three methods are executed on all Workers to implement partial aggregation. The global aggregation operations performed on the Worker where Aggregator Owner is located are described as follows:

i. Implementation of **merge()**

```
@Override
public void merge(KmeansAggrValue value, KmeansAggrValue partial)
    throws IOException {
    value.sums.add(partial.sums);
    value.counts.add(partial.counts);
}
```

In the preceding example, the implementation logic of merge is to sum up the values of sums and counts aggregated by each Worker.

ii. Implementation of terminate()

```
@Override
public boolean terminate(WorkerContext context, KmeansAggrValue value)
    throws IOException {
    // Calculate the new means to be the centroids (original sums)
    DenseMatrix newCentriods = calculateNewCentroids(value.sums, value.counts, value.centroi
ds);
    // print old centroids and new centroids for debugging
    System.out.println("\nsuperstep: " + context.getSuperstep() +
                    "\nold centriod:\n" + value.centroids + " new centriod:\n" + newCentriods);
    boolean converged = isConverged(newCentriods, value.centroids, 0.05d);
    System.out.println("superstep: " + context.getSuperstep() + "/"
                    + (context.getMaxIteration() - 1) + " converged: " + converged);
    if (converged || context.getSuperstep() == context.getMaxIteration() - 1) {
        // converged or reach max iteration, output centriods
        for (int i = 0; i < newCentriods.numRows(); i++) {
            Writable[] centriod = new Writable[newCentriods.numColumns()];
            for (int j = 0; j < newCentriods.numColumns(); j++) {
                centriod[j] = new DoubleWritable(newCentriods.get(i, j));
            }
            context.write(centriod);
        }
        // true means to terminate iteration
        return true;
    }
    // update centriods
    value.centroids.set(newCentriods);
    // false means to continue iteration
    return false;
}
```

In the preceding example, terminate() calls calculateNewCentroids() based on sums and counts to calculate the average value and obtain a new center. isConverged() is then called to determine whether the center is converged based on the Euclidean distance between the new and old centers. If the number of convergences or iterations reaches the upper limit, the new center is generated, and true is returned to end the iteration. Otherwise, the center is updated, and false is returned to continue the iteration.

iii. main method

The main method is used to construct GraphJob, configure related settings, and submit a job.

```
public static void main(String[] args) throws IOException {
    if (args.length < 2)
        printUsage();
    GraphJob job = new GraphJob();
    job.setGraphLoaderClass(KmeansReader.class);
    job.setRuntimePartitioning(false);
    job.setVertexClass(KmeansVertex.class);
    job.setAggregatorClass(KmeansAggregator.class);
    job.addInput(TableInfo.builder().tableName(args[0]).build());
    job.addOutput(TableInfo.builder().tableName(args[1]).build());
    // default max iteration is 30
    job.setMaxIteration(30);
    if (args.length >= 3)
        job.setMaxIteration(Integer.parseInt(args[2]));
    long start = System.currentTimeMillis();
    job.run();
    System.out.println("Job Finished in "
                    + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
}
```

> **Note** If job.setRuntimePartitioning(false) is set to false, data loaded by each Worker is not partitioned based on Partitioner. Data is maintained by the Worker that loads it.

Summary

The basic steps of Aggregator are as follows:

1. When each Worker starts, it executes createStartupValue to create AggregatorValue.
2. Before each iteration starts, each Worker executes createInitialValue to initialize AggregatorValue for the iteration.
3. In an iteration, each vertex uses context.aggregate() to call aggregate() to implement partial iteration in the Worker.
4. Each Worker sends the partial iteration result to the Worker where Aggregator Owner is located.
5. The Worker where Aggregator Owner is located executes merge multiple times to implement global aggregation.
6. The Worker where Aggregator Owner is located executes terminate to process the global aggregation result and determines whether to end the iteration.

1.9.2. Graph feature overview

1.9.2.1. Run a job

The MaxCompute client provides a jar command for running MaxCompute Graph jobs. This command is used in the same way as the jar command in MapReduce.

Command syntax:

```
Usage: jar [<GENERIC_OPTIONS>] <MAIN_CLASS> [ARGS]
-conf <configuration_file> Specify an application configuration file
-classpath <local_file_list> classpaths used to run mainClass
-D <name>=<value> Property value pair, which will be used to run mainClass
-local Run job in local mode
-resources <resource_name_list> file/table resources used in graph, seperate by comma
```

The following table describes the parameters.

Parameters

Parameter	Description
-conf <configuration file>	Indicates a JobConf file.
-classpath <local_file_list>	Indicates the classpath for local execution. It specifies the local paths (including relative path and absolute path) of the JAR package where the main function is located.
-D <prop_name>=<prop_value>	Indicates the Java attribute of <mainClass> in local execution. You can define multiple attributes.
-local	Runs the MapReduce job locally, mainly for program debugging.
-resources <resource_name_list>	Declares the resources used for running the Graph job. You typically need to specify the name of the resource where the Graph job is located in resource_name_list. **Notice** If you read other MaxCompute resources while running the Graph job, you also need to add those resource names to resource_name_list. Multiple resources must be separated by commas (,). If you need to use resources of another project, add the prefix PROJECT_NAME/resources/, for example, -resources otherproject/resources/resfile.

Note The preceding optional parameters are included in <GENERIC_OPTIONS>.

You can directly run the main function in the Graph job to submit the job to MaxCompute, instead of submitting the job through the MaxCompute client. Take the PageRank algorithm as an example.

Example:

```
public static void main(String[] args) throws IOException {
if (args.length < 2)
printUsage();
GraphJob job = new GraphJob();
job.setGraphLoaderClass(PageRankVertexReader.class);
job.setVertexClass(PageRankVertex.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
// Add the resources used in the job to cache resource. These resources correspond to those specifie
d by -resources and libjars in the jar command.
job.addCacheResource("mapreduce-examples.jar");
// Add the used JAR file and other files to the class cache resource. These resources correspond to th
ose specified by -libjars in the jar command.
job.addCacheResourceToClassPath("mapreduce-examples.jar");
// Set the configuration item corresponding to odps_config.ini in the client. Replace it with the actual o
ne in your configuration file.
Account account = new AliyunAccount(accessId, accessKey);
Odps odps = new Odps(account);
odps.setDefaultProject(project);
odps.setEndpoint(endpoint);
SessionState.get().setOdps(odps);
SessionState.get().setLocalRun(false); // default max iteration is 30
job.setMaxIteration(30);
if (args.length >= 3)
job.setMaxIteration(Integer.parseInt(args[2]));
long startTime = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in "
+ (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
```

1.9.2.2. Input and output

The input and output of MaxCompute Graph jobs must be tables. You cannot customize the input or output format.

Job input definition:

```
GraphJob job = new GraphJob();
job.addInput(TableInfo.builder().tableName("tblname").build());
// Tables are used as input.
job.addInput(TableInfo.builder().tableName("tblname").partSpec("pt1=a/pt2=b").build());
// Partitions are used as input.
job.addInput(TableInfo.builder().tableName("tblname").partSpec("pt1=a/pt2=b").build(), new String[]{"
col2", "col0 "});
// Read only col2 and col0 of the input table. Use record.get(0) to obtain col2 in the load() method of G
raphLoader. Both are read in the same sequence.
```

Note

- Multiple inputs are supported.
- Partition filtering is not supported. For more application limits, see Application limits.
- For more information about job input definition, see the addInput method description in GraphJob.
- The framework reads records from the input table and transfers the records to the user-defined GraphLoader to load graph data.

Job output definition:

```
GraphJob job = new GraphJob();
job.addOutput(TableInfo.builder().tableName("table_name").partSpec("pt1=a/pt2=b").build());
// If the output table is a partitioned table, the last level of partitions must be provided.
job.addOutput(TableInfo.builder().tableName("table_name").partSpec("pt1=a/pt2=b").lable("output1")
.build(), true);
// True indicates that the code will overwrite partitions specified in tableinfo, which is similar to the IN
SERT OVERWRITE operation. False is similar to the INSERT INTO operation.
```

Note

- Multiple outputs are supported, and each output is identified by a label.
- A Graph job can use the Write method of WorkContext to write data to an output table during runtime. Multiple outputs must be labeled.
- For more information about job output definition, see the addOutput method description in GraphJob.

1.9.2.3. Read data from resources

1.9.2.3.1. Add resource in Graph program

In addition to the jar command, the following two methods of GraphJob can be used to specify the resources read by Graph:

```
void addCacheResources(String resourceNames)
void addCacheResourcesToClassPath(String resourceNames)
```

1.9.2.3.2. Use resources in Graph

In Graph, you can use the following methods of the corresponding context object WorkerContext to read resources:

```
public byte[] readCacheFile(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName) throws IOException;
public Iterable<byte[]> readCacheArchive(String resourceName, String relativePath)throws IOExceptio
n;
public Iterable<WritableRecord> readResourceTable(String resourceName);
public BufferedInputStream readCacheFileAsStream(String resourceName) throws IOException;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName) throws IOExc
eption;
public Iterable<BufferedInputStream> readCacheArchiveAsStream(String resourceName, String relativ
ePath) throws IOException;
```

Note

- Normally, resources are read in the setup of WorkerComputer, saved in WorkerValue, and obtained through getWorkerValue.
- The preceding stream API is recommended while reading and processing to reduce memory consumption.
- For more information about limits, see Application limits.

1.9.3. Graph SDK introduction

Major APIs

API	Description
GraphJob	GraphJob is inherited from JobConf to define, submit, and manage a MaxCompute Graph job.
Vertex	Vertex is an abstract of a graph and has the following attributes: id, value, halted, and edges. It is implemented through the setVertexClass API in GraphJob.
Edge	Edge is an abstract of a graph and has the following attributes: destVertexId and value. The graph data structure is maintained by an adjacency list. The outgoing edges of a vertex are stored in its edges attribute.
GraphLoader	GraphLoader is used to load graphs. It is set through the setGraphLoaderClass API in GraphJob.

API	Description
VertexResolver	VertexResolver is used to customize the collision processing logic of the revising graph topology. It provides this logic through the setLoadingVertexResolverClass and setComputingVertexResolverClass APIs in GraphJob for graph loading and iteration computing.
Partitioner	Partitioner is used to partition graphs for partition computing. It is set through the setPartitionerClass API in GraphJob. By default, the HashPartitioner is used to first obtain the Vertex ID Hash value, and then to model the number of Workers.
WorkerComputer	WorkerComputer allows customized logic to be executed while Worker starts and exits. WorkerComputer is set through the setWorkerComputerClass API in GraphJob.
Aggregator	Allows you to define one or multiple Aggregators through the setAggregatorClass(Class ...) API in Aggregator.
Combiner	You can set Combiner through the setCombinerClass API in Combiner.
Counters	In the job operating logic, counters can be taken and counted through the WorkerContext API, and the framework will automatically summarize them.
WorkerContext	Context objects encapsulate the functions provided by the framework, such as revising the topology of graphs, sending messages, writing results, reading resources, and so on.

1.9.4. Development and debugging

1.9.4.1. Development procedure

MaxCompute does not provide plug-ins for Graph development. Instead, you can develop MaxCompute Graph programs in Eclipse. The recommended development procedure is as follows:

1. Write Graph code and perform local debugging to test basic functions.
2. Perform cluster debugging to verify results.

1.9.4.2. Development example

Context

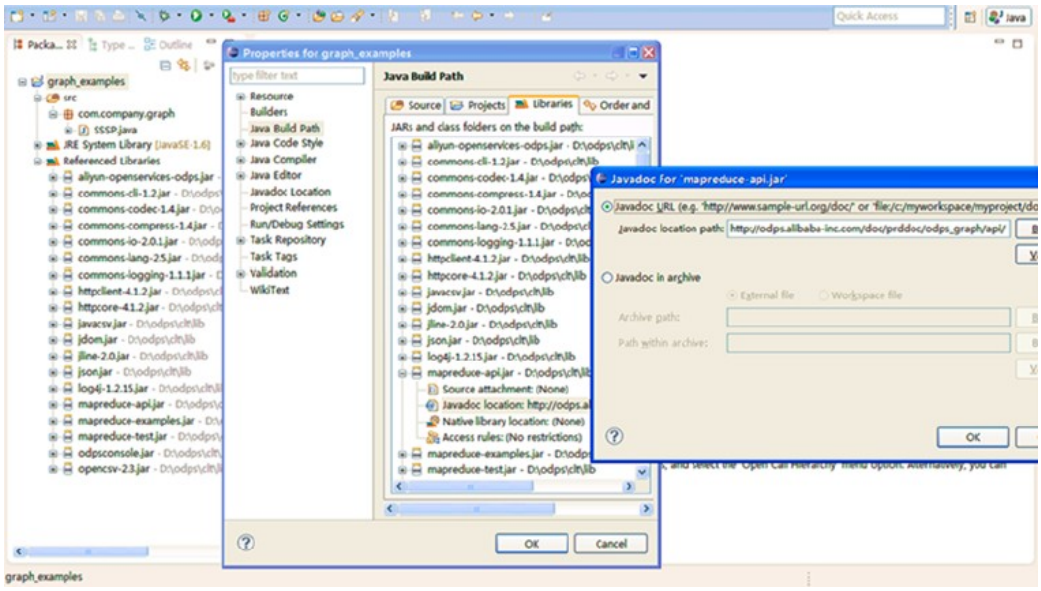
This topic uses the SSSP algorithm as an example to describe how to use Eclipse to develop and debug a Graph program. The development procedure is as follows:

Procedure

1. Create a Java project.

   In this example, the project is graph_examples. Add the JAR package in the lib directory of the MaxCompute client to Build Path of the Eclipse project. The following figure shows a configured Eclipse project.

Create a Java project



2. Develop a MaxCompute Graph program.

   In the actual development process, you can copy a sample program (such as SSSP) and then modify it as required. In this example, only the package path is changed to *package com.aliyun.odps.graph.example*.

3. Compile and build the package. In an Eclipse environment, right-click the source code directory (the src directory in the figure) and choose **Export** > **Java** > **JAR file** to generate a JAR package. Select the path for storing the target JAR package, such as *D:\odps\clt\odps-graph-example-sssp.jar*.

4. Use the MaxCompute client to run SSSP. For more information, see Compile and run a Graph job.

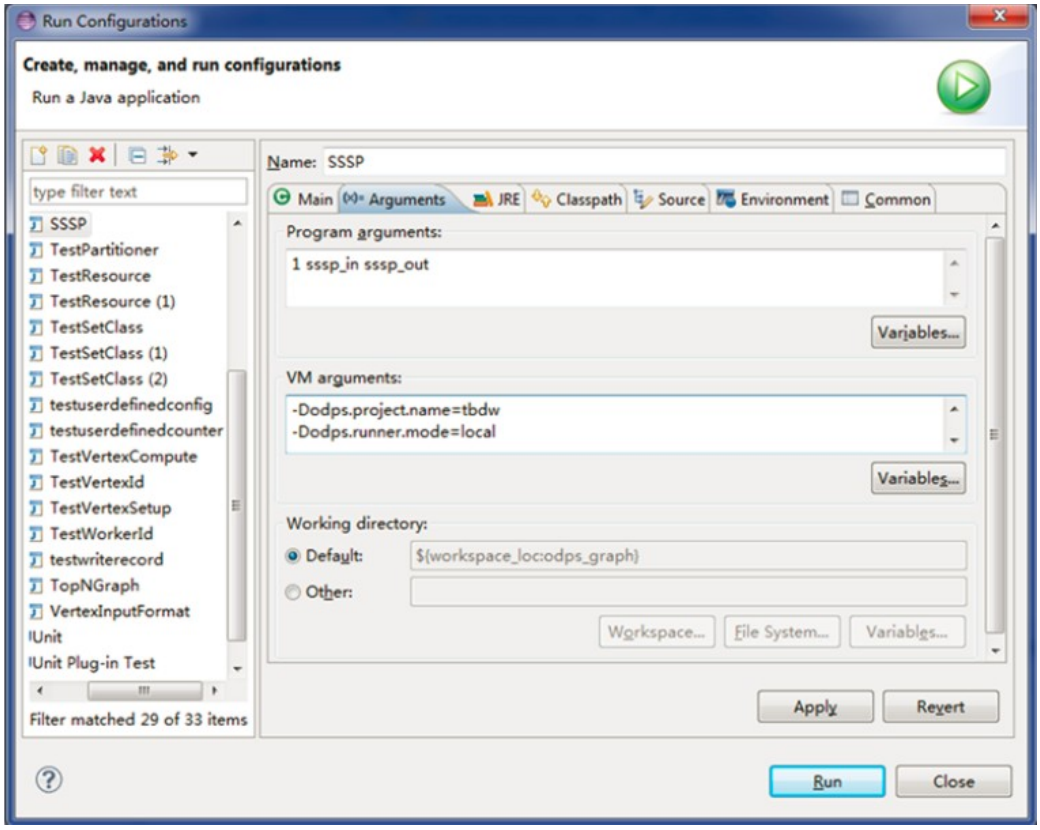
1.9.4.3. Local debugging

MaxCompute Graph supports the local debugging mode. You can use Eclipse for breakpoint debugging. The breakpoint debugging procedure is as follows:

Procedure

1. Select an Eclipse project. Right-click the Graph job main program file (the file that contains the main function), and configure its execution arguments, as shown in the following figure.

   Local debugging



2. On the **Arguments** tab page, set the following arguments as the input parameters of the main program:
   - Program arguments: 1 sssp_in sssp_out.
   - VM arguments: Dodps.runner.mode=local, Dodps.project.name=<project.name>, Dodps.end.point=<end.point>, Dodps.access.id=<access.id>, and Dodps.access.key=<access.key>.
   - For the local mode (odps.end.point not specified), you need to create the sssp_in and sssp_out tables in the warehouse, and add the following data to sssp_in:

```
1,"2:2,3:1,4:4"
2,"1:2,3:2,4:1"
3,"1:1,2:2,5:1"
4,"1:4,2:1,5:1"
5,"3:1,4:1"
```

   > **Note** For more information about the warehouse, see Run MapReduce tasks locally.

3. Click **Run** to run SSSP on the local machine.

   Refer to conf/odps_config.ini in the MaxCompute client for the settings of common parameters. The other parameters are described as follows:
   - odps.runner.mode: The value is local. It is required for the local debugging feature.
   - odps.project.name: specifies the current project, which is required.

- odps.end.point: specifies the current MaxCompute service address, which is optional. If this parameter is not specified, SSSP only reads metadata and data from the tables or resources in the warehouse. If the data does not exist, an error is returned. If this parameter is specified, SSSP first reads data from the warehouse. If the data does not exist, it reads data from the remote MaxCompute service.
- odps.access.id: specifies the AccessKey ID for accessing the MaxCompute service. It is valid only if odps.end.point is specified.
- odps.access.key: specifies the AccessKey secret for accessing the MaxCompute service. It is effective only if odps.end.point is specified.
- odps.cache.resources: specifies the resource list to be used. This parameter is the same as -resources of the jar command.
- odps.local.warehouse: specifies the local warehouse path. It is ./warehouse by default.

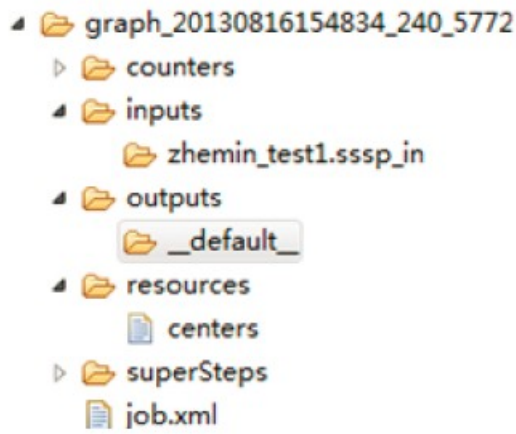
The output is as follows:

```
Counters: 3
com.aliyun.odps.graph.local.COUNTER
TASK_INPUT_BYTE=211
TASK_INPUT_RECORD=5
TASK_OUTPUT_BYTE=161
TASK_OUTPUT_RECORD=5
graph task finish
```

Notice In the preceding examples, the local warehouse must contain the sssp_in and sssp_out tables. For more information about sssp_in and sssp_out, see Compile and run Graph.

1.9.4.4. Temporary directory for local jobs

Each time MaxCompute Graph runs a local debugging job, it creates a temporary directory in the Eclipse project directory, as shown in the following figure.



The temporary directory for a local Graph job contains the following directories and files:

- counters: stores the counter information that is generated during the running of the job.

- inputs: stores the input data of the job. Graph first reads data from the local warehouse. If no data is found, Graph uses the MaxCompute SDK to read data from the server (if odps.end.point is configured). An input operation reads 10 records by default. You can use the Dodps.mapred.local.record.limit parameter to modify the number of records read during each input operation. Up to 10,000 records can be read each time.
- outputs: stores the output data of the job. If there is an output table in the local warehouse, the results in outputs will overwrite data in that table after the job is executed.
- resources: stores the resources used by the job. Similarly, Graph first reads data from the local warehouse. If no data is found, Graph uses the MaxCompute SDK to read data from the server (if odps.end.point is configured).
- job.xml: stores job configurations.
- superstep: stores persistent message information from each iteration.

> **Notice** If you need to output detailed logs during local debugging, place the log4j configuration file named log4j.properties_odps_graph_cluster_debug in the src directory.

1.9.4.5. Cluster debugging

After you perform local debugging, you can perform the following steps to submit a job for cluster testing:

1. Configure the MaxCompute client.
2. Run the `add jar /path/work.jar -f;` command to update the JAR package.
3. Run a JAR command to execute the job and check the operation log and command output.

> **Notice** For more information about running a Graph job in a cluster, see Compile and run a Graph job.

1.9.4.6. Performance optimization

1.9.4.6.1. Configure job parameters

The following table lists the GraphJob configuration parameters that affect job performance.

GraphJob configuration parameters

Parameter	Description
setSplitSize(long)	Indicates the split size in MB. The value must be greater than 0. Default value: 64.
setNumWorkers(int)	Indicates the number of job workers. Value range: 1 to 1,000. Default value: 1. The number of workers is determined by the number of input bytes and split size.
setWorkerCPU(int)	Indicates the CPU resources for a map job. 100 resources are equivalent to one CPU core. Value range: 50 to 800. Default value: 200.

Parameter	Description
setWorkerMemory(int)	Indicates the memory resources in MB for a map job. Value range: 256M to 12G. Default value: 4,096.
setMaxIteration(int)	Indicates the maximum number of iterations. Default value: -1. If the value is equal to or smaller than 0, the job is not terminated after the maximum number of iterations.
setJobPriority(int)	Indicates the job priority. Value range: 0 to 9. Default value: 9. A greater value indicates a lower priority.

Recommendations:

1. Use setNumWorkers to increase the number of workers.
2. Use setSplitSize to reduce the split size and increase the data loading speed.
3. Use setWorkerCPU or setWorkerMemory to increase the CPU or memory resources for workers.
4. Use setMaxIteration to set the maximum number of iterations. For applications that do not require precise results, you can reduce the number of iterations to accelerate the iterating process.

Use setNumWorkers and setSplitSize together to accelerate data loading. If setNumWorkers is workerNum, setSplitSize is splitSize, and the total number of input bytes is inputSize, splitNum equals inputSize divided by splitSize. The relationship between workerNum and splitNum is as follows:

1. If splitNum is equal to workerNum, each worker loads one split.
2. If splitNum is greater than workerNum, each worker loads one or more splits.
3. If splitNum is smaller than workerNum, each worker loads zero or one split.

Therefore, you can adjust workerNum and splitSize to obtain a suitable loading speed. In the first two cases, data loading is faster. In the iteration phase, you only need to adjust workerNum. If you set runtime partitioning to false, we recommend that you either use setSplitSize to adjust the number of workers, or ensure the conditions in the first two cases are met. In the third case, the number of vertices in some of the workers is 0. In this case, you can use set odps.graph.split.size=<m>; set odps.graph.worker.num=<n>; before the JAR command, which achieves the same effect as setNumWorkers and setSplitSize.

Another common performance problem is data skew. As indicated by the counters, some workers process much more vertices or edges than others.

Data skew usually occurs when the number of vertices, edges, or messages corresponding to certain keys is far greater than that corresponding to other keys. These keys are distributed for processing by a small number of workers, resulting in long execution time of these workers. You can use the following methods to resolve this issue:

- Use Combiner to aggregate the messages of vertices corresponding to the keys, to reduce the number of generated messages.
- Improve the business logic.

1.9.4.6.2. Use Combiner

Developers can set Combiner to reduce the memory and network traffic consumed by message storage, and reduce job execution time. For more information, see the Combiner description in Graph SDK overview.

1.9.4.6.3. Reduce data input

If a disk stores large volumes of data, reading data from the disk may prolong the processing time. You can reduce the number of bytes to be read to increase the overall throughput and improve job performance. You can use either of the following methods:

- Reduce data input: For some decision-making applications, processing sampled data only affects the precision of the results, not the overall accuracy. In this case, you can sample specific data to the input table for further processing.
- Avoid reading unnecessary fields: The TableInfo class provided in the MaxCompute Graph framework can read specific columns (sent through a column name array), instead of the entire table or partition. This effectively reduces the amount of input data and improves job performance.

1.9.4.6.4. JAR packages

The following JAR packages are loaded on the JVM that runs Graph programs by default. You do not need to manually upload these resources, or use -libjars to specify them in a command.

- commons-codec-1.3.jar
- commons-io-2.0.1.jar
- commons-lang-2.5.jar
- commons-logging-1.0.4.jar
- commons-logging-api-1.0.4.jar
- guava-14.0.jar
- json.jar
- log4j-1.2.15.jar
- slf4j-api-1.4.3.jar
- slf4j-log4j12-1.4.3.jar
- xmlenc-0.52.jar

> **Notice** In CLASSPATH of the running JVM, the preceding JAR packages are loaded before your JAR package. This may cause version conflicts. For example, your program calls a certain class function of commons-codec-1.5.jar, but the function is not included in the current MaxCompute packages. In this case, you can choose to call a similar function in version 1.3 or wait until MaxCompute is upgraded to the required version.

1.9.5. Application limits

- Each job can reference up to 256 resources. Each table or archive is considered as one unit.
- The total resource size referenced by a job cannot exceed 512 MB.
- Each job can have up to 1,024 inputs (the number of input tables cannot exceed 64). Each job can have up to 256 outputs.
- Labels specified for multiple outputs cannot be null or empty strings. The label length cannot

exceed 256, and the label can contain only upper-case letters (A to Z), lower-case letters (a to z), digits (0 to 9), underlines (_), pound signs (#), periods (.), and hyphens (-).

- The number of custom counters in a job cannot exceed 64. The counter group name and counter name cannot contain pound signs (#), and the total length of both names cannot exceed 100.
- The number of workers for each job is calculated by the framework. The maximum number of workers is 1,000. An error is returned when the number of workers exceeds this value.
- Each worker consumes 200 units of CPU resources by default. The range of resources consumed is 50 to 800.
- Each worker consumes 4,096 MB memory by default. The range of memory consumed is 256 MB to 12 GB.
- Each worker can read from a single resource up to 64 times.
- The split size is 64 MB by default, but can be defined by the user. The split size must be greater than 0, and the maximum value is in the range of 20 to 9223372036854775807.
- GraphLoader/Vertex/Aggregator in MaxCompute Graph are restricted by Java Sandbox (however, the main program of a Graph job is not subject to this restriction) while they run in a cluster. For more information, see Java sandbox restrictions.

1.9.6. Sample programs

1.9.6.1. SSSP

Dijkstra's algorithm is a typical algorithm for calculating the Single Source Shortest Path (SSSP) in a directed graph.

Shortest path: For a weighted directed graph G = (V, E), many paths are available from source vertex s to sink vertex v. The path with the smallest sum of edge weights is called the shortest path from s to v. The algorithm is implemented as follows:

- Initialization: The distance from s to s is 0 (d[s] = 0), and the distance from u to s is infinite (d[u] = ∞).
- Iteration: If an edge from u to v exists, the shortest distance from s to v is updated to d[v] = min(d[v], d[u] + weight(u, v)). The iteration ends until the distance from all vertices to s does not change.

> **Note** The implementation process determines that the algorithm is applicable to the MaxCompute Graph program. Each vertex maintains the current shortest distance to the source vertex. If the value changes, a message containing the new value and the edge weight is sent to the adjacent vertices. In the next iteration, the adjacent vertices update the current shortest distance based on the received message. The iteration ends when the current shortest distance values of all vertices do not change.

Example:

```
import java.io.IOException;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.Combiner;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.Edge;
```

```
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.data.TableInfo;
public class SSSP {
public static final String START_VERTEX = "sssp.start.vertex.id";
/**Define SSSPVertex, where:
* The vertex value indicates the current shortest distance from this vertex to source vertex startVerte
xId.
* The compute() method uses the iteration formula d[v] = min(d[v], d[u] + weight(u, v)) to update the v
ertex value.
* The cleanup() method writes the vertex and its shortest distance to the source vertex to the result t
able.
 **/
public static class SSSPVertex extends
Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
private static long startVertexId = -1;
public SSSPVertex() {
this.setValue(new LongWritable(Long.MAX_VALUE));
}
public boolean isStartVertex(
ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context) {
if (startVertexId == -1) {
String s = context.getConfiguration().get(START_VERTEX);
startVertexId = Long.parseLong(s);
}
return getId().get() == startVertexId;
}
@Override
public void compute(
ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context, Iterable<LongWri
table> messages) throws IOException {
long minDist = isStartVertex(context) ? 0 : Integer.MAX_VALUE;
for (LongWritable msg : messages) { if (msg.get() < minDist) {
minDist = msg.get();
}
}
if (minDist < this.getValue().get()) {
```

```
this.setValue(new LongWritable(minDist));
if (hasEdges()) {
for (Edge<LongWritable, LongWritable> e : this.getEdges()) {
context.sendMessage(e.getDestVertexId(), new LongWritable(minDist + e.getValue().get()));
}
}
} else {
voteToHalt();
// If the vertex value does not change, voteToHalt() is called to notify the framework that this vertex
enters the halted state. The calculation ends when all vertices enter the halted state.
}
}
@Override
public void cleanup(
WorkerContext<LongWritable, LongWritable, LongWritable, LongWritable> context) throws IOExceptio
n {
context.write(getId(), getValue());
}
}
/** Define MinLongCombiner and combine messages sent to the same vertex to optimize performance
and reduce memory usage.**/
public static class MinLongCombiner extends
Combiner<LongWritable, LongWritable> {
@Override
public void combine(LongWritable vertexId, LongWritable combinedMessage, LongWritable messageTo
Combine) throws IOException {
if (combinedMessage.get() > messageToCombine.get()) {
combinedMessage.set(messageToCombine.get());
}
}
}
/** Define the SSSPVertexReader class, load a graph, and parse each record in the table into a vertex.
The first column of the record is the vertex ID, and the second column stores all edge sets starting fro
m the vertex, such as 2:2,3:1,4:4.**/
public static class SSSPVertexReader extends
GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
@Override
public void load(LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context) throws IOExcepti
on {
```

```
SSSPVertex vertex = new SSSPVertex();
vertex.setId((LongWritable) record.get(0));
String[] edges = record.get(1).toString().split(",");
for (int i = 0; i < edges.length; i++) {
String[] ss = edges[i].split(":");
vertex.addEdge(new LongWritable(Long.parseLong(ss[0])), new LongWritable(Long.parseLong(ss[1]))
);
}
context.addVertexRequest(vertex);
}
}
public static void main(String[] args) throws IOException { if (args.length < 2) {
System.out.println("Usage: <startnode> <input> <output>");
System.exit(-1);
}
GraphJob job = new GraphJob();
// Define GraphJob, specify the implementation of Vertex/GraphLoader/Combiner, and specify input an
d output tables.
job.setGraphLoaderClass(SSSPVertexReader.class);
job.setVertexClass(SSSPVertex.class);
job.setCombinerClass(MinLongCombiner.class);
job.set(START_VERTEX, args[0]);
job.addInput(TableInfo.builder().tableName(args[1]).build());
job.addOutput(TableInfo.builder().tableName(args[2]).build());
long startTime = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.2. PageRank

PageRank is an algorithm for Web page ranking. For more information, see PageRank. The input of the algorithm is a digraph G, where Vertex represents pages. If there is a link between page A to page B, there is an Edge linking A and B. Basic principles of the algorithm are as follows:

- Initialization: Vertex value means rank value of PageRank (double type). Initially, the value of all Vertices is 1/TotalNumVertices.
- Iteration formula: PageRank(i)=0.15/TotalNumVertices+0.85*sum. Sum indicates the sum of PageRank(j)/out_degree(j). (j indicates all vertices pointing to vertex i.)

Note The PageRank algorithm is best suited to be run on MaxCompute Graph as each j point maintains its PageRank value and sends PageRank(j)/out_degree(j) to its adjacent vertex (to vote it) per iteration. Upon the next iteration, each vertex re-calculates the PageRank value using the iteration formula.

Example:

```
import java.io.IOException;
import org.apache.log4j.Logger;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
public class PageRank {
private final static Logger LOG = Logger.getLogger(PageRank.class);
/**
* Defines PageRankVertex, where:
* The vertex value indicates the current PageRank value of the vertex (web page).
* The compute() method uses the iteration formula PageRank(i)=0.15/TotalNumVertices+0.85*sum to u
pdate the vertex value.
* The cleanup() method writes the vertex and its PageRank value to the result table.
**/
public static class PageRankVertex extends
Vertex<Text, DoubleWritable, NullWritable, DoubleWritable> {
@Override
public void compute(
ComputeContext<Text, DoubleWritable, NullWritable, DoubleWritable> context, Iterable<DoubleWritabl
e> messages) throws IOException {
if (context.getSuperstep() == 0) {
setValue(new DoubleWritable(1.0 / context.getTotalNumVertices()));
} else if (context.getSuperstep() >= 1) { double sum = 0;
for (DoubleWritable msg : messages) { sum += msg.get();
}
```

```
}
DoubleWritable vertexValue = new DoubleWritable( (0.15f / context.getTotalNumVertices()) + 0.85f * s
um);
setValue(vertexValue);
}
if (hasEdges()) {
context.sendMessageToNeighbors(this, new DoubleWritable(getValue()
.get() / getEdges().size()));
}
}
@Override
public void cleanup(
WorkerContext<Text, DoubleWritable, NullWritable, DoubleWritable> context) throws IOException {
context.write(getId(), getValue());
}
}
/** Define the PageRankVertexReader class, load a graph, and resolve each record in the table into a v
ertex. The first column of the record is the start vertex and other columns are the destination vertices
.**/
public static class PageRankVertexReader extends
GraphLoader<Text, DoubleWritable, NullWritable, DoubleWritable> {
@Override public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<Text, DoubleWritable, NullWritable, DoubleWritable> context) throws IOException {
PageRankVertex vertex = new PageRankVertex();
vertex.setValue(new DoubleWritable(0));
vertex.setId((Text) record.get(0));
System.out.println(record.get(0));
for (int i = 1; i < record.size(); i++) {
Writable edge = record.get(i);
System.out.println(edge.toString());
if (!( edge.equals(NullWritable.get()))) {
vertex.addEdge(new Text(edge.toString()), NullWritable.get());
}
}
LOG.info("vertex edgs size: " + (vertex.hasEdges() ? vertex.getEdges().size() : 0));
context.addVertexRequest(vertex);
}
}
private static void printUsage() {
System.out.println("Usage: <in> <out> [Max iterations (default 30)]");
```

```
System.exit(-1);
}
public static void main(String[] args) throws IOException { if (args.length < 2)
printUsage();
GraphJob job = new GraphJob();
// Define GraphJob and specify the implementation method of Vertex/GraphLoader, the maximum num
ber of iterations (> 30 by default), and input and output tables.
job.setGraphLoaderClass(PageRankVertexReader.class);
job.setVertexClass(PageRankVertex.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
// default max iteration is 30
job.setMaxIteration(30); if (args.length >= 3)
job.setMaxIteration(Integer.parseInt(args[2]));
long startTime = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in "
+ (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.3. K-means clustering

K-means clustering is a basic macro-clustering algorithm. Basic principles of the K-means clustering algorithm are as follows: Clustering is performed around k points in space, and the closest vertices are classified. The values of the clustering centers are successively updated through iterations until the optimal clustering result is obtained.

Assuming the sample set is divided into k sets or categories, the steps in the algorithm are as follows:

- Select initial center of k classes.
- In the ith iteration, select any sample, solve its path to k center, and then classify the sample into the class of shortest path to center.
- Use the mean method to update the center value of the class.
- For all k clustering centers, if the value remains unchanged or is less than a certain threshold after iteration of the first two steps, the iteration ends. Otherwise, the iteration continues.

Example:

```
import java.io.DataInput; import java.io.DataOutput;
import java.io.IOException;
importorg.apache.log4j.Logger;
import com.aliyun.odps.io.WritableRecord;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
```

```
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
public class Kmeans {
private final static Logger LOG = Logger.getLogger(Kmeans.class);
/**Define KmeansVertex. The compute() method is simple. It calls the aggregate() method of the conte
xt object and transmits the value of the current vertex (in Tuple type and expressed by vector).**/
public static class KmeansVertex extends
Vertex<Text, Tuple, NullWritable, NullWritable> {
@Override
public void compute(
ComputeContext<Text, Tuple, NullWritable, NullWritable> context, Iterable<NullWritable> messages) t
hrows IOException { context.aggregate(getValue());
}
}
/** Define the KmeansVertexReader class, load a graph, and parse each record in the table as a vertex
. The vertex ID does not matter, and transmitted recordNum is used as the ID. The vertex value is the T
uple consisting of all columns of the record.**/
public static class KmeansVertexReader extends
GraphLoader<Text, Tuple, NullWritable, NullWritable> {
@Override
public void load(LongWritable recordNum, WritableRecord record, MutationContext<Text, Tuple, NullWr
itable, NullWritable> context) throws IOException {
KmeansVertex vertex = new KmeansVertex();
vertex.setId(new Text(String.valueOf(recordNum.get())));
vertex.setValue(new Tuple(record.getAll()));
context.addVertexRequest(vertex);
}
}
public static class KmeansAggrValue implements Writable {
Tuple centers = new Tuple();
Tuple sums = new Tuple();
```

```
Tuple counts = new Tuple();
@Override
public void write(DataOutput out) throws IOException {
centers.write(out);
sums.write(out); counts.write(out);
}
@Override
public void readFields(DataInput in) throws IOException {
centers = new Tuple();
centers.readFields(in);
sums = new Tuple();
sums.readFields(in);
counts = new Tuple();
counts.readFields(in);
}
@Override
public String toString() {
return "centers " + centers.toString() + ", sums " + sums.toString()
+ ", counts " + counts.toString();
}
}
/**
* Defines KmeansAggregator. This class encapsulates the main logic of the Kmeans algorithm, where,
* createInitialValue creates an initial value for each iteration (k-class center point). In first iteration (s
uperstep equals to 0), the value is the initial center point. Otherwise, the value is the new center poin
t when the last iteration ends.
* The aggregate() method calculates the distance from each vertex to centers of different classes, cla
ssifies the vertex as the class of the nearest center, and updates sum and count of the class.
* The merge() method combines sums and counts collected by each Worker.
* The terminate() method calculates the new central point based on the sum and count of each class. I
f the distance between the new and old central points is less than a threshold value or the number of
iterations reaches the upper limit, the iteration ends (false is returned). The final central point is writt
en to the resulting table.
**/
public static class KmeansAggregator extends Aggregator<KmeansAggrValue> {
@SuppressWarnings("rawtypes")
@Override
public KmeansAggrValue createInitialValue(WorkerContext context)
throws IOException {
KmeansAggrValue aggrVal = null;
if (context.getSuperstep() == 0) {
```

```
aggrVal = new KmeansAggrValue();
aggrVal.centers = new Tuple();
aggrVal.sums = new Tuple();
aggrVal.counts = new Tuple();
byte[] centers = context.readCacheFile("centers");
String lines[] = new String(centers).split("\n");
for (int i = 0;
i < lines.length; i++) { String[] ss = lines[i].split(",");
Tuple center = new Tuple();
Tuple sum = new Tuple();
for (int j = 0; j < ss.length; ++j) {
center.append(new DoubleWritable(Double.valueOf(ss[j].trim())));
sum.append(new DoubleWritable(0.0));
}
LongWritable count = new LongWritable(0);
aggrVal.sums.append(sum); aggrVal.counts.append(count);
aggrVal.centers.append(center);
}
} else {
aggrVal = (KmeansAggrValue) context.getLastAggregatedValue(0);
}
return aggrVal;
}
@Override
Public void aggregate (glasvalue, object item ){
int min = 0;
double mindist = Double.MAX_VALUE;
Tuple point = (Tuple) item;
for (int i = 0;
i < value.centers.size();
i++) { Tuple center = (Tuple) value.centers.get(i);
// use Euclidean Distance, no need to calculate sqrt
double dist = 0.0d;
for (int j = 0; j < center.size(); j++) {
double v = ((DoubleWritable) point.get(j)).get()
- ((DoubleWritable) center.get(j)).get();
dist += v * v;
}
if (dist < mindist) { mindist = dist; min = i;
}
```

```
}
// update sum and count
Tuple sum = (Tuple) value.sums.get(min);
for (int i = 0;
i < point.size(); i++) {
DoubleWritable s = (DoubleWritable) sum.get(i); s.set(s.get() + ((DoubleWritable) point.get(i)).get());
}
LongWritable count = (LongWritable) value.counts.get(min);
count.set(count.get() + 1);
}
@Override
public void merge(KmeansAggrValue value, KmeansAggrValue partial) {
for (int i = 0; i < value.sums.size(); i++) {
Tuple sum = (Tuple) value.sums.get(i);
Tuple that = (Tuple) partial.sums.get(i);
for (int j = 0; j < sum.size(); j++) {
DoubleWritable s = (DoubleWritable) sum.get(j);
s.set(s.get() + ((DoubleWritable) that.get(j)).get());
}
}
for (int i = 0; i < value.counts.size(); i++) {
LongWritable count = (LongWritable) value.counts.get(i);
count.set(count.get() + ((LongWritable) partial.counts.get(i)).get());
}
}
@SuppressWarnings("rawtypes")
@Override
public boolean terminate(WorkerContext context, KmeansAggrValue value) throws IOException {
// compute new centers
Tuple newCenters = new Tuple(value.sums.size());
for (int i = 0; i < value.sums.size(); i++) {
Tuple sum = (Tuple) value.sums.get(i);
Tuple newCenter = new Tuple(sum.size());
LongWritable c = (LongWritable) value.counts.get(i);
for (int j = 0; j < sum.size(); j++) {
DoubleWritable s = (DoubleWritable) sum.get(j);
double val = s.get() / c.get();
newCenter.set(j, new DoubleWritable(val));
// reset sum for next iteration
s.set(0.0d);
}
```

```
// reset count for next iteration
c.set(0);
newCenters.set(i, newCenter);
}
// update centers
Tuple oldCenters = value.centers; value.centers = newCenters;
LOG.info("old centers: " + oldCenters + ", new centers: " + newCenters);
// compare new/old centers
boolean converged = true;
for (int i = 0; i < value.centers.size() && converged; i++) {
Tuple oldCenter = (Tuple) oldCenters.get(i);
Tuple newCenter = (Tuple) newCenters.get(i); double sum = 0.0d;
for (int j = 0; j < newCenter.size(); j++) {
double v = ((DoubleWritable) newCenter.get(j)).get() - ((DoubleWritable) oldCenter.get(j)).get();
sum += v * v;
}
double dist = Math.sqrt(sum);
LOG.info("old center: " + oldCenter + ", new center: " + newCenter + ", dist: " + dist);
// converge threshold for each center: 0.05
converged = dist < 0.05d;
}
if (converged || context.getSuperstep() == context.getMaxIteration() - 1) {
// converged or reach max iteration, output centers
for (int i = 0; i < value.centers.size(); i++) { context.write(((Tuple) value.centers.get(i)).toArray());
}
// true means to terminate iteration
return true;
}
// false means to continue iteration
return false;
}
}
private static void printUsage() {
System.out.println("Usage: <in> <out> [Max iterations (default 30)]");
System.exit(-1);
}
/**Define GraphJob, and specify the implementation method of Vertex, GraphLoader, or Aggregator, th
e maximum number of iterations (30 by default), and input and output tables. **/
public static void main(String[] args) throws IOException {
if (args.length < 2)
printUsage();
```

```
printUsage();
GraphJob job = new GraphJob();
job.setGraphLoaderClass(KmeansVertexReader.class);
job.setRuntimePartitioning(false);
// Specify job.setRuntimePartitioning(false). For the K-means algorithm, vertices do not need to be dis
tributed during graph loading. If RuntimePartitioning is set to false, the performance for graph loading
is improved.
job.setVertexClass(KmeansVertex.class);
job.setAggregatorClass(KmeansAggregator.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
// default max iteration is 30
job.setMaxIteration(30); if (args.length >= 3)
job.setMaxIteration(Integer.parseInt(args[2]));
long start = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
}
}
```

1.9.6.4. BiPartiteMatching

In a bipartite graph, all vertices can be divided into two sets, to which the two vertices of each edge respectively belong. For a bipartite graph G, M is its subgraph. If any two edges of M's edge set are not attached to the same vertex, then M is a match. The bipartite graph matching is usually used for information matching in scenarios with clear supply and demand relationships (such as online dating websites).

The procedure is as follows:

- Start from the first vertex on the left, select unmatched vertex to search for augmented path.
- If it goes through an unmatched vertex, the search is successful.
- Update path information, match number of Edge +1, and stop searching.
- If no augmented path is found, it does not search again from the specific vertex.

Example:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.Random;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.WorkerContext;
```

```
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Text;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
public class BipartiteMatching {
private static final Text UNMATCHED = new Text("UNMATCHED");
public static class TextPair implements Writable {
public Text first; public Text second;
public TextPair() { first = new Text();
second = new Text();
}
public TextPair(Text first, Text second) {
this.first = new Text(first);
this.second = new Text(second);
}
@Override
public void write(DataOutput out) throws IOException {
first.write(out);
second.write(out);
}
@Override
public void readFields(DataInput in) throws IOException {
first = new Text();
first.readFields(in);
second = new Text();
second.readFields(in);
}
@Override
public String toString() { return first + ": " + second;
}
}
public static class BipartiteMatchingVertexReader extends
GraphLoader<Text, TextPair, NullWritable, Text> {
@Override
public void load(LongWritable recordNum, WritableRecord record, MutationContext<Text, TextPair, Null
Writable, Text> context) throws IOException {
BipartiteMatchingVertex vertex = new BipartiteMatchingVertex();
vertex.setId((Text) record.get(0));
```

```
vertex.setValue(new TextPair(UNMATCHED, (Text) record.get(1)));
String[] adjs = record.get(2).toString().split(",");
for (String adj:adjs) {
vertex.addEdge(new Text(adj), null);
}
context.addVertexRequest(vertex);
}
}
public static class BipartiteMatchingVertex extends
Vertex<Text, TextPair, NullWritable, Text> {
private static final Text LEFT = new Text("LEFT");
private static final Text RIGHT = new Text("RIGHT");
private static Random rand = new Random();
@Override
public void compute(
ComputeContext<Text, TextPair, NullWritable, Text> context, Iterable<Text> messages) throws IOExce
ption {
if (isMatched()) { voteToHalt();
return;
}
switch ((int) context.getSuperstep() % 4) {
case 0:
if (isLeft()) {
context.sendMessageToNeighbors(this, getId());
}
break;
case 1:
if (isRight()) {
Text luckyLeft = null;
for (Text message : messages) { if (luckyLeft == null) {
luckyLeft = new Text(message);
} else {
if (rand.nextInt(1) == 0) { luckyLeft.set(message);
}
}
}
if (luckyLeft ! = null) { context.sendMessage(luckyLeft, getId());
}
}
break;
case 2:
```

```
case 2:
if (isLeft()) {
Text luckyRight = null;
for (Text msg : messages) { if (luckyRight == null) {
luckyRight = new Text(msg);
} else {
if (rand.nextInt(1) == 0) { luckyRight.set(msg);
}
}
}
if (luckyRight ! = null) {
setMatchVertex(luckyRight);
context.sendMessage(luckyRight, getId());
}
}
break; case 3:
if (isRight()) {
for (Text msg : messages) { setMatchVertex(msg);
}
}
break;
}
}
@Override
public void cleanup(
WorkerContext<Text, TextPair, NullWritable, Text> context) throws IOException {
context.write(getId(), getValue().first);
}
private boolean isMatched() {
return ! getValue().first.equals(UNMATCHED);
}
private boolean isLeft() {
return getValue().second.equals(LEFT);
}
private boolean isRight() {
return getValue().second.equals(RIGHT);
}
private void setMatchVertex(Text matchVertex) { getValue().first.set(matchVertex);
}
}
private static void printUsage() {
```

```
System.err.println("BipartiteMatching <input> <output> [maxIteration]");
}
public static void main(String[] args) throws IOException { if (args.length < 2) {
printUsage();
}
GraphJob job = new GraphJob();
job.setGraphLoaderClass(BipartiteMatchingVertexReader.class);
job.setVertexClass(BipartiteMatchingVertex.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
int maxIteration = 30;
if (args.length > 2) {
maxIteration = Integer.parseInt(args[2]);
}
job.setMaxIteration(maxIteration);
job.run();
}
}
```

1.9.6.5. Strongly-connected component

A directed graph is called a strongly-connected graph if every vertex is reachable from every other vertex. A strongly-connected sub-graph with a large number of vertices in a directed graph is called a strongly-connected component. This algorithm example is based on the parallel coloring algorithm.

Each vertex contains the following two parts:

- colorID: stores the color of the vertex (v) during forward traversal. At the end of computing, the vertices with the same colorID belong to one strongly-connected component.
- transposeNeighbors: stores neighbor IDs of v in the transpose graph of the input graph.

The algorithm is implemented as follows:

- Transpose graph formation: contains two supersteps. In the first superstep, each vertex sends a message with its ID to all its outgoing neighbors. These IDs are stored in transposeNeighbors in the second superstep.
- Triming: contains one superstep. Each vertex with only one incoming or outgoing edge sets its colorID to its own ID, and becomes inactive. Subsequent messages sent to these vertexes are ignored.
- Forward traversal: contains two subphases (supersteps): Start and Rest. In the Start phase, each vertex sets its colorID to its own ID, and sends the ID to outgoing neighbors. In the Rest phase, each vertex uses the maximum colorID it received to update its own colorID, and propagates the colorID until the colorIDs converge. When the colorIDs converge, the master process sets the phase to backward traversal.
- Backward traversal: contains two subphases, Start and Rest. In the Start phase, each vertex whose ID equals its colorID propagates its ID to the vertices in transposeNeighbors and sets its

status as inactive. Subsequent messages sent to these vertexes are ignored. In each of the Rest phase supersteps, each vertex receives a message matching its colorID, propagates its colorID in the transpose graph, and sets its status as inactive. If there are still active vertices after this step, the process goes back to the trimming phase.

Example:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.BooleanWritable;
import com.aliyun.odps.io.IntWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
* Definition from Wikipedia:
* In the mathematical theory of directed graphs, a graph is said
* to be strongly connected if every vertex is reachable from every
* other vertex. The strongly connected components of an arbitrary
* directed graph form a partition into subgraphs that are themselves
* strongly connected.
*
* Algorithms with four phases as follows.
* 1. Transpose Graph Formation: Requires two supersteps. In the first
* superstep, each vertex sends a message with its ID to all its > outgoing
* neighbors, which in the second superstep are stored > in transposeNeighbors.
*
* 2. Trimming: Takes one superstep. Every vertex with only in-coming or
* only outgoing edges (or neither) sets its colorID to its own ID and
* becomes inactive. Messages subsequently sent to the vertex > are ignored.
*
* 3. Forward-Traversal: There are two sub phases: Start and Rest. In the
```

```
* Start phase, each vertex sets its colorID to its own ID and > propagates
* its ID to its outgoing neighbors. In the Rest phase, vertices update
* their own colorIDs with the minimum colorID they have seen, and > propagate
* their colorIDs, if updated, until the colorIDs converge.
* Set the phase to Backward-Traversal when the colorIDs converge.
*
* 4. Backward-Traversal: We again break the phase into Start and Rest.
* In Start, every vertex whose ID equals its colorID propagates its ID > to
* the vertices in transposeNeighbors and sets itself inactive. > Messages
* subsequently sent to the vertex are ignored. In each of the Rest > phase supersteps,
* each vertex receiving a message that matches its colorID: (1) > propagates
* its colorID in the transpose graph; (2) sets itself inactive. > Messages
* subsequently sent to the vertex are ignored. Set the phase back to Trimming
* if not all vertex are inactive.
*
* http://ilpubs.stanford.edu:8090/1077/3/p535-salihoglu.pdf
**/
public class StronglyConnectedComponents {
public final static int STAGE_TRANSPOSE_1 = 0;
public final static int STAGE_TRANSPOSE_2 = 1;
public final static int STAGE_TRIMMING = 2;
public final static int STAGE_FW_START = 3;
public final static int STAGE_FW_REST = 4;
public final static int STAGE_BW_START = 5;
public final static int STAGE_BW_REST = 6;
/**
* The value is composed of component id, incoming neighbors,
* active status and updated status.
**/
public static class MyValue implements Writable {
LongWritable sccID;// strongly connected component id
Tuple inNeighbors; // transpose neighbors
BooleanWritable active; // vertex is active or not
BooleanWritable updated; // sccID is updated or not
public MyValue() {
this.sccID = new LongWritable(Long.MAX_VALUE);
this.inNeighbors = new Tuple();
this.active = new BooleanWritable(true);
this.updated = new BooleanWritable(false);
}
```

```
public void setSccID(LongWritable sccID) {
this.sccID = sccID;
}
public LongWritable getSccID() {
return this.sccID;
}
public void setInNeighbors(Tuple inNeighbors) {
this.inNeighbors = inNeighbors;
}
public Tuple getInNeighbors() {
return this.inNeighbors;
}
public void addInNeighbor(LongWritable neighbor) {
this.inNeighbors.append(new LongWritable(neighbor.get()));
}
public boolean isActive() {
return this.active.get();
}
public void setActive(boolean status) {
this.active.set(status);
}
public boolean isUpdated() {
return this.updated.get();
}
public void setUpdated(boolean update) {
this.updated.set(update);
}
@Override
public void write(DataOutput out) throws IOException {
this.sccID.write(out);
this.inNeighbors.write(out);
this.active.write(out);
this.updated.write(out);
}
@Override
public void readFields(DataInput in) throws IOException {
this.sccID.readFields(in);
this.inNeighbors.readFields(in);
this.active.readFields(in);
this.updated.readFields(in);
}
```

```
@Override
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append("sccID: " + sccID.get());
sb.append(" inNeighbores: " + inNeighbors.toDelimitedString(','));
sb.append(" active: " + active.get());
sb.append(" updated: " + updated.get());
return sb.toString();
}
}
public static class SCCVertex extends
Vertex<LongWritable, MyValue, NullWritable, LongWritable> {
public SCCVertex() {
this.setValue(new MyValue());
}
@Override
public void compute(
ComputeContext<LongWritable, MyValue, NullWritable, LongWritable> context, Iterable<LongWritable>
msgs) throws IOException {
// Messages sent to inactive vertex are ignored.
if (! this.getValue().isActive()) {
this.voteToHalt(); return;
}
int stage = ((SCCAggrValue)context.getLastAggregatedValue(0)).getStage(); switch (stage) {
case STAGE_TRANSPOSE_1:
context.sendMessageToNeighbors(this, this.getId());
break;
case STAGE_TRANSPOSE_2:
for (LongWritable msg: msgs) {
this.getValue().addInNeighbor(msg);
}
case STAGE_TRIMMING:
this.getValue().setSccID(getId());
if (this.getValue().getInNeighbors().size() == 0 || this.getNumEdges() == 0) {
this.getValue().setActive(false);
}
break;
case STAGE_FW_START: this.getValue().setSccID(getId());
context.sendMessageToNeighbors(this, this.getValue().getSccID());
break;
case STAGE_FW_REST:
```

```
long minSccID = Long.MAX_VALUE;
for (LongWritable msg : msgs) {
if (msg.get() < minSccID) { minSccID = msg.get();
}
}
if (minSccID < this.getValue().getSccID().get()) {
this.getValue().setSccID(new LongWritable(minSccID));
context.sendMessageToNeighbors(this, this.getValue().getSccID());
this.getValue().setUpdated(true);
} else {
this.getValue().setUpdated(false);
}
break;
case STAGE_BW_START:
if (this.getId().equals(this.getValue().getSccID())) {
for (Writable neighbor : this.getValue().getInNeighbors().getAll()) {
context.sendMessage((LongWritable)neighbor, this.getValue().getSccID());
}
this.getValue().setActive(false);
}
break;
case STAGE_BW_REST: this.getValue().setUpdated(false);
for (LongWritable msg : msgs) {
if (msg.equals(this.getValue().getSccID())) {
for (Writable neighbor : this.getValue().getInNeighbors().getAll()) {
context.sendMessage((LongWritable)neighbor, this.getValue().getSccID());
}
this.getValue().setActive(false);
this.getValue().setUpdated(true);
break;
}
}
break;
}
context.aggregate(0, getValue());
}
@Override
public void cleanup(
WorkerContext<LongWritable, MyValue, NullWritable, LongWritable> context)
throws IOException {
```

```
context.write(getId(), getValue().getSccID());
}
}
/**
* The SCCAggrValue maintains global stage and graph updated and > active status.
* updated is true only if one vertex is updated.
* active is true only if one vertex is active.
*/
public static class SCCAggrValue implements Writable {
IntWritable stage = new IntWritable(STAGE_TRANSPOSE_1);
BooleanWritable updated = new BooleanWritable(false);
BooleanWritable active = new BooleanWritable(false);
public void setStage(int stage) { this.stage.set(stage);
}
public int getStage() { return this.stage.get();
}
public void setUpdated(boolean updated) {
this.updated.set(updated);
}
public boolean getUpdated() {
return this.updated.get();
}
public void setActive(boolean active) {
this.active.set(active);
}
public boolean getActive() {
return this.active.get();
}
@Override
public void write(DataOutput out) throws IOException {
this.stage.write(out);
this.updated.write(out);
this.active.write(out);
}
@Override
public void readFields(DataInput in) throws IOException {
this.stage.readFields(in);
this.updated.readFields(in);
this.active.readFields(in);
}
}
```

```
/**
* The job of SCCAggregator is to schedule global stage in > every superstep.
*/
public static class SCCAggregator extends Aggregator<SCCAggrValue> {
@SuppressWarnings("rawtypes")
@Override
public SCCAggrValue createStartupValue(WorkerContext context) throws IOException { return new SCC
AggrValue();
}
@SuppressWarnings("rawtypes")
@Override
public SCCAggrValue createInitialValue(WorkerContext context) throws IOException {
return (SCCAggrValue) context.getLastAggregatedValue(0);
}
@Override
public void aggregate(SCCAggrValue value, Object item) throws IOException { MyValue v = (MyValue)it
em;
if ((value.getStage() == STAGE_FW_REST || value.getStage() == STAGE_BW_REST)&& v.isUpdated()) { va
lue.setUpdated(true);
}
// only active vertex invoke aggregate()
value.setActive(true);
}
@Override
public void merge(SCCAggrValue value, SCCAggrValue partial) throws IOException {
boolean updated = value.getUpdated() || partial.getUpdated();
value.setUpdated(updated);
boolean active = value.getActive() || partial.getActive();
value.setActive(active);
}
@SuppressWarnings("rawtypes")
@Override
public boolean terminate(WorkerContext context, SCCAggrValue value) throws IOException {
// If all vertices is inactive, job is over.
if (! value.getActive()) { return true;
}
// state machine
switch (value.getStage()) {
case STAGE_TRANSPOSE_1:value.setStage(STAGE_TRANSPOSE_2);
break;
case STAGE_TRANSPOSE_2:value.setStage(STAGE_TRIMMING);
```

```
case STAGE_TRANSPOSE_2:value.setStage(STAGE_TRIMMING);
break;
case STAGE_TRIMMING:value.setStage(STAGE_FW_START);
break;
case STAGE_FW_START: value.setStage(STAGE_FW_REST);
break;
case STAGE_FW_REST:if (value.getUpdated()) {
value.setStage(STAGE_FW_REST);
} else {
value.setStage(STAGE_BW_START);
}
break;
case STAGE_BW_START: value.setStage(STAGE_BW_REST);
break;
case STAGE_BW_REST:if (value.getUpdated()) { value.setStage(STAGE_BW_REST);
} else { value.setStage(STAGE_TRIMMING);
}
break;
}
value.setActive(false);
value.setUpdated(false);
return false;
}
}
public static class SCCVertexReader extends
GraphLoader<LongWritable, MyValue, NullWritable, LongWritable> {
@Override public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, MyValue, NullWritable, LongWritable> context) throws IOException {
SCCVertex vertex = new SCCVertex();
vertex.setId((LongWritable) record.get(0));
String[] edges = record.get(1).toString().split(",");
for (int i = 0; i < edges.length; i++) { try {
long destID = Long.parseLong(edges[i]);
vertex.addEdge(new LongWritable(destID), NullWritable.get());
} catch(NumberFormatException nfe) { System.err.println("Ignore " + nfe);
}
}
context.addVertexRequest(vertex);
}
}
```

```
public static void main(String[] args) throws Exception {
if (args.length < 2) {
System.out.println("Usage: <input> <output>");
System.exit(-1);
}
GraphJob job = new GraphJob();
job.setGraphLoaderClass(SCCVertexReader.class);
job.setVertexClass(SCCVertex.class);
job.setAggregatorClass(SCCAggregator.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
long startTime = System.currentTimeMillis();
job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.6. Connected component

Two vertices are connected if a path exists between them. Undirected graph G is called a connected graph if every two vertices in the graph are connected. Otherwise, G is called an unconnected graph. A connected sub-graph with a large number of vertices is called a connected component. This algorithm calculates connected component members of each vertex, and outputs the connected component of the vertex value that includes the smallest vertex ID. The smallest vertex ID is propagated along edges to all vertices of the connected component.

Example:

```
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.graph.examples.SSSP.MinLongCombiner;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.WritableRecord;
/**
* Compute the connected component membership of each vertex and output
* each vertex which's value containing the smallest id in the > connected
```

```
* component containing that vertex.
*
* Algorithm: propagate the smallest vertex id along the edges to all
* vertices of a connected component.
*
*/
public class ConnectedComponents {
public static class CCVertex extends
Vertex<LongWritable, LongWritable, NullWritable, LongWritable> {
@Override
public void compute(
ComputeContext<LongWritable, LongWritable, NullWritable, LongWritable> context, Iterable<LongWrit
able> msgs) throws IOException {
if (context.getSuperstep() == 0L) {
this.setValue(getId());
context.sendMessageToNeighbors(this, getValue());
return;
}
long minID = Long.MAX_VALUE;
for (LongWritable id : msgs) {
if (id.get() < minID) { minID = id.get();
}
}
if (minID < this.getValue().get()) {
this.setValue(new LongWritable(minID));
context.sendMessageToNeighbors(this, getValue());
} else {
this.voteToHalt();
}
}
/**
* Output Table Description:
* +-----------------+----------------------------------------+
* Field | Type | Comment |
* +-----------------+----------------------------------------+
* v | bigint | vertex id |
* minID | bigint | smallest id in the connected component |
* +-----------------+----------------------------------------+
*/
@Override
```

```
public void cleanup(
WorkerContext<LongWritable, LongWritable, NullWritable, LongWritable> context) throws IOException
{
context.write(getId(), getValue());
}
}
/**
* Input Table Description:
* +-----------------+-----------------------------------------------------+
* Field | Type | Comment |
* +-----------------+-----------------------------------------------------+
* v | bigint | vertex id |
* es | string | comma separated target vertex id of outgoing edges |
* +-----------------+-----------------------------------------------------+
*
* Example:
* For graph:
* 1 ----- 2
* | |
* 3 ----- 4
* Input table:
* +------------+
* v | es |
* +------------+
* | 1 | 2,3 |
* | 2 | 1,4 |
* | 3 | 1,4 |
* | 4 | 2,3 |
* +------------+
*/
public static class CCVertexReader extends
GraphLoader<LongWritable, LongWritable, NullWritable, LongWritable> {
@Override
public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, LongWritable, NullWritable, LongWritable> context) throws IOExceptio
n {
CCVertex vertex = new CCVertex();
vertex.setId((LongWritable) record.get(0));
String[] edges = record.get(1).toString().split(",");
for (int i = 0; i < edges.length; i++) {
```

```
long destID = Long.parseLong(edges[i]);
vertex.addEdge(new LongWritable(destID), NullWritable.get());
}
context.addVertexRequest(vertex);
}
}
public static void main(String[] args) throws IOException {
if (args.length < 2) {
System.out.println("Usage: <input> <output>");
System.exit(-1);
}
GraphJob job = new GraphJob();
job.setGraphLoaderClass(CCVertexReader.class);
job.setVertexClass(CCVertex.class);
job.setCombinerClass(MinLongCombiner.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
long startTime = System.currentTimeMillis();
job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.7. Topological sorting

For a directed edge (u,v), all vertex sequences satisfying u < v are called topological sequences. Topological sorting is an algorithm that is used to calculate the topological sequence of a directed graph.

The algorithm is implemented as follows:

- A vertex without incoming edges in the graph is found and output.
- The output vertex and all its outgoing edges are deleted.
- The preceding steps are repeated until all vertices are output.

Example:

```
import java.io.IOException;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.Combiner;
import com.aliyun.odps.graph.ComputeContext;
```

```
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.BooleanWritable;
import com.aliyun.odps.io.WritableRecord;
public class TopologySort {
private final static Log LOG = LogFactory.getLog(TopologySort.class);
public static class TopologySortVertex extends
Vertex<LongWritable, LongWritable, NullWritable, LongWritable> {
@Override
public void compute(
ComputeContext<LongWritable, LongWritable, NullWritable, LongWritable> context, Iterable<LongWrit
able> messages) throws IOException {
// in superstep 0, each vertex sends message whose value is 1 to its
// neighbors
if (context.getSuperstep() == 0) { if (hasEdges()) {
context.sendMessageToNeighbors(this, new LongWritable(1L));
}
} else if (context.getSuperstep() >= 1) {
// compute each vertex's indegree
long indegree = getValue().get();
for (LongWritable msg : messages) {
indegree += msg.get();
}
setValue(new LongWritable(indegree));
if (indegree == 0) {
voteToHalt();
if (hasEdges()) {
context.sendMessageToNeighbors(this, new LongWritable(-1L));
}
context.write(new LongWritable(context.getSuperstep()), getId());
LOG.info("vertex: " + getId());
}
context.aggregate(new LongWritable(indegree));
}
}
}
```

```
public static class TopologySortVertexReader extends
GraphLoader<LongWritable, LongWritable, NullWritable, LongWritable> {
@Override public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, LongWritable, NullWritable, LongWritable> context) throws IOExceptio
n {
TopologySortVertex vertex = new TopologySortVertex();
vertex.setId((LongWritable) record.get(0));
vertex.setValue(new LongWritable(0));
String[] edges = record.get(1).toString().split(",");
for (int i = 0; i < edges.length; i++) {
long edge = Long.parseLong(edges[i]);
if (edge >= 0) {
vertex.addEdge(new LongWritable(Long.parseLong(edges[i])), NullWritable.get());
}
}
LOG.info(record.toString());
context.addVertexRequest(vertex);
}
}
public static class LongSumCombiner extends
Combiner<LongWritable, LongWritable> {
@Override
public void combine(LongWritable vertexId, LongWritable combinedMessage, LongWritable messageTo
Combine) throws IOException {
combinedMessage.set(combinedMessage.get() + messageToCombine.get());
}
}
public static class TopologySortAggregator extends
Aggregator<BooleanWritable> {
@SuppressWarnings("rawtypes")
@Override
public BooleanWritable createInitialValue(WorkerContext context) throws IOException {
return new BooleanWritable(true);
}
@Override
public void aggregate(BooleanWritable value, Object item) throws IOException {
boolean hasCycle = value.get();
boolean inDegreeNotZero = ((LongWritable) item).get() == 0 ? false : true;
value.set(hasCycle && inDegreeNotZero);
}
```

```
}
@Override
public void merge(BooleanWritable value, BooleanWritable partial) throws IOException {
value.set(value.get() && partial.get());
}
@SuppressWarnings("rawtypes")
@Override
public boolean terminate(WorkerContext context, BooleanWritable value) throws IOException {
if (context.getSuperstep() == 0) {
// since the initial aggregator value is true, and in superstep we don't
// do aggregate
return false;
}
return value.get();
}
}
public static void main(String[] args) throws IOException { if (args.length ! = 2) {
System.out.println("Usage : <inputTable> <outputTable>");
System.exit(-1);
}
// Input format
// 0 1, 2
// 1 3
// 2 3
// 3 -1
// The first column is vertexid, and the second column is the destination vertexid of the vertex. If the v
alue is -1, the vertex does not have any outgoing edges.
// Output format
// 0 0
// 1 1
// 1 2
// 2 3
// The first column is the supstep value, in which the topological sequence is hidden. The second colu
mn is vertexid.
// TopologySortAggregator is used to determine if the graph has any loops.
// If the input graph has a loop, the iteration ends when the indegree of all active vertices is not 0.
// You can use records in the input and output tables to determine if the graph has loops.
GraphJob job = new GraphJob();
job.setGraphLoaderClass(TopologySortVertexReader.class);
job.setVertexClass(TopologySortVertex.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
```

```
job.addOutput(TableInfo.builder().tableName(args[1]).build());
job.setCombinerClass(LongSumCombiner.class);
job.setAggregatorClass(TopologySortAggregator.class);
long startTime = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.8. Linear regression

In statistics, linear regression is a statistical analysis method used to determine the dependency between two or more variables. compared with the classification algorithm that predicts discrete data, the regression algorithm can predict continuous value-type data. The linear regression algorithm defines the loss function as the sum of the least square errors of a sample set. It solves the weight vector by minimizing the loss function.

A common solution is the gradient descent method. It is implemented as follows:

- Initialize the weight vector to provide the descent speed and iterations (or iteration convergence condition).
- Calculate the least square error for each sample.
- Calculate the sum of the least square error, and update the weight based on the descent speed.
- Repeat iterations until convergence occurs.

Example:

```
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.Aggregator;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.io.DoubleWritable;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
```

```
* LineRegression input: y,x1,x2,x3,.....
*
* @author shiwan.ch
* @update jiasheng.tjs running parameters are like: tjs_lr_in > tjs_lr_out 1500 2
* 0.07
*/
public class LinearRegression {
public static class GradientWritable implements Writable {
Tuple lastTheta;
Tuple currentTheta;
Tuple tmpGradient;
LongWritable count;
DoubleWritable lost;
@Override
public void readFields(DataInput in) throws IOException {
lastTheta = new Tuple();
lastTheta.readFields(in);
currentTheta = new Tuple();
currentTheta.readFields(in);
tmpGradient = new Tuple();
tmpGradient.readFields(in);
count = new LongWritable();
count.readFields(in);
/* update 1: add a variable to store lost at every iteration */
lost = new DoubleWritable();
lost.readFields(in);
}
@Override
public void write(DataOutput out) throws IOException {
lastTheta.write(out);
currentTheta.write(out);
tmpGradient.write(out);
count.write(out);
lost.write(out);
}
}
public static class LinearRegressionVertex extends
Vertex<LongWritable, Tuple, NullWritable, NullWritable> {
@Override
public void compute(
ComputeContext<LongWritable, Tuple, NullWritable, NullWritable> context, Iterable<NullWritable> mes
```

```
sages) throws IOException {
context.aggregate(getValue());
}
}
public static class LinearRegressionVertexReader extends
GraphLoader<LongWritable, Tuple, NullWritable, NullWritable> {
@Override
public void load(LongWritable recordNum, WritableRecord record, MutationContext<LongWritable, Tupl
e, NullWritable, NullWritable> context)
throws IOException {
LinearRegressionVertex vertex = new LinearRegressionVertex();
vertex.setId(recordNum);
vertex.setValue(new Tuple(record.getAll())); context.addVertexRequest(vertex);
}
}
public static class LinearRegressionAggregator extends
Aggregator<GradientWritable> {
@SuppressWarnings("rawtypes")
@Override
public GradientWritable createInitialValue(WorkerContext context) throws IOException {
if (context.getSuperstep() == 0) {
/* set initial value, all 0 */
GradientWritable grad = new GradientWritable();
grad.lastTheta = new Tuple();
grad.currentTheta = new Tuple();
grad.tmpGradient = new Tuple();
grad.count = new LongWritable(1);
grad.lost = new DoubleWritable(0.0);
int n = (int) Long.parseLong(context.getConfiguration().get("Dimension"));
for (int i = 0; i < n; i++) { grad.lastTheta.append(new DoubleWritable(0));
grad.currentTheta.append(new DoubleWritable(0));
grad.tmpGradient.append(new DoubleWritable(0));
}
return grad;
} else
return (GradientWritable) context.getLastAggregatedValue(0);
}
public static double vecMul(Tuple value, Tuple theta) {
/* perform this partial computing: y(i)-hθ(x(i)) for each sample */
/* value denote a piece of sample and value(0) is y */
double sum = 0.0;
```

```
double sum = 0.0;
for (int j = 1; j < value.size(); j++)
sum += Double.parseDouble(value.get(j).toString()) * Double.parseDouble(theta.get(j).toString());
Double tmp = Double.parseDouble(theta.get(0).toString()) + sum -Double.parseDouble(value.get(0).to
String());
return tmp;
}
@Override
public void aggregate(GradientWritable gradient, Object value) throws IOException {
/*
* perform on each vertex--each sample i:set theta(j) for each sample > i
* for each dimension
*/
double tmpVar = vecMul((Tuple) value, gradient.currentTheta);
/*
* update 2:local worker aggregate(), perform like merge() below. This
* means the variable gradient denotes the previous aggregated value
*/
gradient.tmpGradient.set(0, new DoubleWritable( ((DoubleWritable) gradient.tmpGradient.get(0)).get(
) + tmpVar));
gradient.lost.set(Math.pow(tmpVar, 2));
/*
* calculate (y(i)-hθ(x(i)))x(i)(j) for each sample i for each
* dimension j
*/
for (int j = 1; j < gradient.tmpGradient.size(); j++) gradient.tmpGradient.set(j, new DoubleWritable(
((DoubleWritable) gradient.tmpGradient.get(j)).get() + tmpVar * Double.parseDouble(((Tuple) value).g
et(j).toString())));
}
@Override
public void merge(GradientWritable gradient, GradientWritable partial) throws IOException {
/* perform SumAll on each dimension for all samples. */
Tuple master = (Tuple) gradient.tmpGradient;
Tuple part = (Tuple) partial.tmpGradient;
for (int j = 0; j < gradient.tmpGradient.size(); j++) {
DoubleWritable s = (DoubleWritable) master.get(j);
s.set(s.get() + ((DoubleWritable) part.get(j)).get());
}
gradient.lost.set(gradient.lost.get() + partial.lost.get());
}
@SuppressWarnings("rawtypes")
```

```
@Override
public boolean terminate(WorkerContext context, GradientWritable gradient) throws IOException {
/*
* 1. calculate new theta 2. judge the diff between last step and this
* step, if smaller than the threshold, stop iteration
*/
gradient.lost = new DoubleWritable(gradient.lost.get()/ (2 * context.getTotalNumVertices()));
/*
* we can calculate lost in order to make sure the algorithm is running > on
* the right direction (for debug)
*/
System.out.println(gradient.count + " lost:" + gradient.lost);
Tuple tmpGradient = gradient.tmpGradient;
System.out.println("tmpGra" + tmpGradient);
Tuple lastTheta = gradient.lastTheta;
Tuple tmpCurrentTheta = new Tuple(gradient.currentTheta.size());
System.out.println(gradient.count + " terminate_start_last:" + lastTheta);
double alpha = 0.07; // learning rate
// alpha =
// Double.parseDouble(context.getConfiguration().get("Alpha"));
/* perform theta(j) = theta(j)-alpha*tmpGradient */
long M = context.getTotalNumVertices();
/*
* update 3: add (/M) on the code. The original code forget this step
*/
for (int j = 0; j < lastTheta.size(); j++) { tmpCurrentTheta
.set( j,
new DoubleWritable(Double.parseDouble(lastTheta.get(j)
.toString()) - alpha / M * Double.parseDouble(tmpGradient.get(j).toString())));
}
System.out.println(gradient.count + " terminate_start_current:" + tmpCurrentTheta);
// judge if convergence is happening.
double diff = 0.00d;
for (int j = 0; j < gradient.currentTheta.size(); j++)
diff += Math.pow(((DoubleWritable) tmpCurrentTheta.get(j)).get() - ((DoubleWritable) lastTheta.get(j)
).get(), 2);
if (
/*
* Math.sqrt(diff) < 0.00000000005d ||
*/
Long.parseLong(context.getConfiguration().get("Max_Iter_Num")) == gradient.count
```

```
.get()) { context.write(gradient.currentTheta.toArray());
return true;
}
gradient.lastTheta = tmpCurrentTheta;
gradient.currentTheta = tmpCurrentTheta;
gradient.count.set(gradient.count.get() + 1);
int n = (int) Long.parseLong(context.getConfiguration().get("Dimension"));
/*
* update 4: Important!!! Remember this step. Graph won't reset the
* initial value for global variables at the beginning of each iteration
*/
for (int i = 0; i < n; i++) {
gradient.tmpGradient.set(i, new DoubleWritable(0));
}
return false;
}
}
public static void main(String[] args) throws IOException { GraphJob job = new GraphJob();
job.setGraphLoaderClass(LinearRegressionVertexReader.class); job.setRuntimePartitioning(false);
job.setNumWorkers(3);
job.setVertexClass(LinearRegressionVertex.class);
job.setAggregatorClass(LinearRegressionAggregator.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
job.setMaxIteration(Integer.parseInt(args[2])); // Numbers of Iteration
job.setInt("Max_Iter_Num", Integer.parseInt(args[2]));
job.setInt("Dimension", Integer.parseInt(args[3])); // Dimension
job.setFloat("Alpha", Float.parseFloat(args[4])); // Learning rate
long start = System.currentTimeMillis(); job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - start) / 1000.0 + " seconds");
}
}
```

1.9.6.9. Count triangles

This algorithm is used to calculate the number of triangles passing through each vertex. The algorithm is implemented as follows:

- Each vertex sends its ID to all outgoing neighbors.
- Each vertex stores information about incoming and outgoing neighbors, and sends this information to outgoing neighbors.

- Each vertex calculates the number of endpoint intersections for each edge, calculates the sum, and outputs the results to a table.
- The number of triangles is the sum of the output results in the table divided by 3.

Example:

```
import java.io.IOException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.WorkerContext;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.NullWritable;
import com.aliyun.odps.io.Tuple;
import com.aliyun.odps.io.Writable;
import com.aliyun.odps.io.WritableRecord;
/**
* Compute the number of triangles passing through each vertex.
*
* The algorithm can be computed in three supersteps:
* I. Each vertex sends a message with its ID to all its outgoing
* neighbors.
* II. The incoming neighbors and outgoing neighbors are stored and
* send to outgoing neighbors.
* III. For each edge compute the intersection of the sets at destination
* vertex and sum them, then output to table.
*
* The triangle count is the sum of output table and divide by three > since
* each triangle is counted three times.
*
**/
public class TriangleCount {
public static class TCVertex extends
Vertex<LongWritable, Tuple, NullWritable, Tuple> {
@Override
public void setup(
WorkerContext<LongWritable, Tuple, NullWritable, Tuple> context) throws IOException {
// collect the outgoing neighbors
```

```
Tuple t = new Tuple();
if (this.hasEdges()) {
for (Edge<LongWritable, NullWritable> edge : this.getEdges()) {
t.append(edge.getDestVertexId());
}
}
this.setValue(t);
}
@Override
public void compute(
ComputeContext<LongWritable, Tuple, NullWritable, Tuple> context, Iterable<Tuple> msgs) throws IOE
xception {
if (context.getSuperstep() == 0L) {
// sends a message with its ID to all its outgoing neighbors
Tuple t = new Tuple(); t.append(getId());
context.sendMessageToNeighbors(this, t);
} else if (context.getSuperstep() == 1L) {
// store the incoming neighbors
for (Tuple msg : msgs) {
for (Writable item : msg.getAll()) {
if (! this.getValue().getAll().contains((LongWritable)item)) {
this.getValue().append((LongWritable)item);
}
}
}
// send both incoming and outgoing neighbors to all outgoing neighbors
context.sendMessageToNeighbors(this, getValue());
} else if (context.getSuperstep() == 2L) {
// count the sum of intersection at each edge
long count = 0;
for (Tuple msg : msgs) {
for (Writable id : msg.getAll()) {
if (getValue().getAll().contains(id)) { count ++;
}
}
}
// output to table
context.write(getId(), new LongWritable(count));
this.voteToHalt();
}
}
```

```
}
public static class TCVertexReader extends
GraphLoader<LongWritable, Tuple, NullWritable, Tuple> {
@Override public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, Tuple, NullWritable, Tuple> context) throws IOException {
TCVertex vertex = new TCVertex();
vertex.setId((LongWritable) record.get(0));
String[] edges = record.get(1).toString().split(",");
for (int i = 0; i < edges.length; i++) { try {
long destID = Long.parseLong(edges[i]);
vertex.addEdge(new LongWritable(destID), NullWritable.get());
} catch(NumberFormatException nfe) { System.err.println("Ignore " + nfe);
}
}
context.addVertexRequest(vertex);
}
}
public static void main(String[] args) throws IOException { if (args.length < 2) {
System.out.println("Usage: <input> <output>"); System.exit(-1);
}
GraphJob job = new GraphJob();
job.setGraphLoaderClass(TCVertexReader.class);
job.setVertexClass(TCVertex.class);
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addOutput(TableInfo.builder().tableName(args[1]).build());
long startTime = System.currentTimeMillis();
job.run();
System.out.println("Job Finished in " + (System.currentTimeMillis() - startTime) / 1000.0 + " seconds");
}
}
```

1.9.6.10. GraphLoader

The following example describes how to compile a graph job program to load data of different types. It mainly covers how GraphLoader and VertexResolver are used together to build the graph.

Example:

```
import java.io.IOException;
import com.aliyun.odps.conf.Configuration;
import com.aliyun.odps.data.TableInfo;
```

```
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.graph.ComputeContext;
import com.aliyun.odps.graph.GraphJob;
import com.aliyun.odps.graph.GraphLoader;
import com.aliyun.odps.graph.Vertex;
import com.aliyun.odps.graph.VertexResolver;
import com.aliyun.odps.graph.MutationContext;
import com.aliyun.odps.graph.VertexChanges;
import com.aliyun.odps.graph.Edge;
import com.aliyun.odps.io.LongWritable;
import com.aliyun.odps.io.WritableComparable;
import com.aliyun.odps.io.WritableRecord;
/**
* A MaxCompute Graph job uses MaxCompute tables as the input. Assume that a job has two input tabl
es, one storing vertices and the other storing edges.
* The format of the table storing vertices is as follows:
* +------------------------+
* VertexID | VertexValue |
* +------------------------+
* | id0| 9|
* +------------------------+
* | id1| 7|
* +------------------------+
* | id2| 8|
* +------------------------+
*
* The format of the table storing edges is as follows:
* +-----------------------------------+
* VertexID | DestVertexID| EdgeValue|
* +-----------------------------------+
* | id0| id1| 1|
* +-----------------------------------+
* | id0| id2| 2|
* +-----------------------------------+
* | id2| id1| 3|
* +-----------------------------------+
*
* The two preceding tables show that id0 has two outgoing edges pointing to id1 and id2. id2 has an o
utgoing edge pointing to id1, and id1 has no outgoing edges.
*
* For data of this type, in GraphLoader::load(LongWritable, Record, MutationContext), > MutationCont
```

```
ext#addVertexRequest(Vertex) can be used to add vertices to the graph, while link MutationContext#
addEdgeRequest(WritableComparable,Edge) can be used to add edges to the graph. In link VertexRes
olver#resolve(WritableComparable, Vertex, VertexChanges, boolean), vertices and edges added in the
load() method are combined to a vertex object, which is used as the returned value and added to the
graph for participating in computation.
*
**/
public class VertexInputFormat {
private final static String EDGE_TABLE = "edge.table";
/**
* Resolve a record to vertices and edges. Each record indicates a vertex or an edge based on its sourc
e.
*
* Enter a record to generate key-value pairs as you process com.aliyun.odps.mapreduce.Mapper#map.
The keys are vertex IDs, and the values are vertices or edges written based on the context. These key
-value pairs are summarized based on vertex IDs using LoadingVertexResolver.
*
* Note: Vertices or edges added here are requests sent based on the record content, and are not use
d in computation. Only vertices or edges added using VertexResolver participate in computation.
**/
public static class VertexInputLoader extends
GraphLoader<LongWritable, LongWritable, LongWritable, LongWritable> {
private boolean isEdgeData;
/**
* Configure VertexInputLoader.
*
* @param conf
* Indicate the configured parameters of a job. These parameters are configured in the MAIN function o
f GraphJob, or set on the console.
* @param workerId
* Indicate the serial number of the operating worker. It starts from 0 and can be used to build a unique
vertex ID.
* @param inputTableInfo
* Indicate information about the input table loaded to the current worker. The information can be used
to determine the type of current input data (record format).
**/
@Override
public void setup(Configuration conf, int workerId, TableInfo inputTableInfo) {
isEdgeData = conf.get(EDGE_TABLE).equals(inputTableInfo.getTableName());
}
/**
```

```
* Based on the record content, resolve corresponding edges and send a request to add them to the gr
aph.
*
* @param recordNum
* Indicate the record serial number, which starts from 1 and is separately counted in each worker.
* @param record
* Indicate the record in the input table. It contains three columns, indicating the first vertex, last verte
x, and edge weight.
* @param context
* Indicate the context for adding resolved edges to the graph.
**/
@Override public void load(
LongWritable recordNum, WritableRecord record,
MutationContext<LongWritable, LongWritable, LongWritable, LongWritable> context) throws IOExcepti
on {
if (isEdgeData) {
/**
* Data comes from the table that stores edges.
*
* 1. The first column indicates the first vertex ID.
**/
LongWritable sourceVertexID = (LongWritable) record.get(0);
/**
* 2. The second column indicates the last vertex ID.
**/
LongWritable destinationVertexID = (LongWritable) record.get(1);
/**
* 3. The third column indicates the edge weight.
**/
LongWritable edgeValue = (LongWritable) record.get(2);
/**
* 4. Create an edge that consists of the last vertex ID and edge weight.
**/
Edge<LongWritable, LongWritable> edge = new Edge<LongWritable, LongWritable>( destinationVertex
ID, edgeValue);
/**
* 5. Send a request to add an edge to the first vertex.
**/
context.addEdgeRequest(sourceVertexID, edge);
/**
```

```
* 6. If each record indicates a bidirectional edge, repeat steps 4 and 5. Edge<LongWritable, > LongWrit
able> edge2 = new
* Edge<LongWritable, LongWritable>( sourceVertexID, edgeValue);
* context.addEdgeRequest(destinationVertexID, edge2);
**/
} else {
/**
* Data comes from the table that stores vertices.
*
* 1. The first column indicates the vertex ID.
**/
LongWritable vertexID = (LongWritable) record.get(0);
/**
* 2. The second column indicates the vertex value.
**/
LongWritable vertexValue = (LongWritable) record.get(1);
/**
* 3. Create a vertex that consists of the vertex ID and vertex value.
**/
MyVertex vertex = new MyVertex();
/**
* 4. Initialize the vertex.
**/
vertex.setId(vertexID); vertex.setValue(vertexValue);
/**
* 5. Send a request for adding a vertex.
**/
context.addVertexRequest(vertex);
}
}
}
/**
* Summarize key-value pairs generated using GraphLoader::load(LongWritable, Record, > MutationCo
ntext), which is similar to
* Reduce in com.aliyun.odps.mapreduce.Reducer. For the unique vertex > ID, all actions such as
* adding or deleting vertices or edges for the ID are stored in VertexChanges.
*
* Note: Not only conflicting vertices or edges added by using the load() method are called. (A conflict o
ccurs when multiple same vertex objects or duplicate edges are added.)
* All IDs requested to be generated using the load() method are called.
**/
```

```
public static class LoadingResolver extends
VertexResolver<LongWritable, LongWritable, LongWritable, LongWritable> {
/**
* Process a request for adding/deleting vertices or edges for an ID.
*
* VertexChanges has four APIs, which correspond to the four APIs of MutationContext:
* VertexChanges::getAddedVertexList() corresponds to
* MutationContext::addVertexRequest(Vertex).
* In the load() method, if vertex objects with the same ID are requested to be added, such vertex objec
ts are collected to the returned list.
* VertexChanges::getAddedEdgeList() corresponds to
* MutationContext::addEdgeRequest(WritableComparable, Edge)
* If edge objects with the same first vertex ID are requested to be added, such edge objects are collec
ted to the returned list.
* VertexChanges::getRemovedVertexCount() corresponds to
* MutationContext::removeVertexRequest(WritableComparable)
* If vertices with the same ID are requested to be deleted, the number of total deletion requests is ret
urned.
* VertexChanges#getRemovedEdgeList() corresponds to
* MutationContext#removeEdgeRequest(WritableComparable, WritableComparable)
* If edge objects with the same first vertex ID are requested to be deleted, such edge objects are colle
cted to the returned list.
*
* By processing ID changes, you can state whether the ID participates in computation using the return
ed value. If the returned vertex is not NULL,
* the ID participates in subsequent computation. If the returned vertex is NULL, the ID does not partici
pate in subsequent computation.
*
* @param vertexId
* Indicate the ID of the vertex to be added, or the ID of the first vertex of the edge to be added.
* @param vertex
* Indicate an existing vertex object. Its value is always NULL in the data loading phase.
* @param vertexChanges
* Indicate the set of vertices or edges to be added/deleted for the ID.
* @param hasMessages
* Indicate whether the ID has any input messages. Its value is always false in the data loading phase.
**/
@Override
public Vertex<LongWritable, LongWritable, LongWritable, LongWritable> resolve( LongWritable vertexI
d,
Vertex<LongWritable, LongWritable, LongWritable, LongWritable> vertex, VertexChanges<LongWritabl
```

```
e, LongWritable, LongWritable, LongWritable> vertexChanges, boolean hasMessages) throws IOExcept
ion {
/**
* 1. Obtain the vertex object for computation.
**/
MyVertex computeVertex = null;
if (vertexChanges.getAddedVertexList() == null
|| vertexChanges.getAddedVertexList().isEmpty()) { computeVertex = new MyVertex(); computeVertex.
setId(vertexId);
} else {
/**
* Each record indicates a unique vertex in the table that stores vertices.
**/
computeVertex = (MyVertex) vertexChanges.getAddedVertexList().get(0);
}
/**
* 2. Add the edge, which is requested to be added to the vertex, to the vertex object. If the data is a p
ossible duplicate, perform deduplication based on the algorithm needs.
**/
if (vertexChanges.getAddedEdgeList() ! = null) {
for (Edge<LongWritable, LongWritable> edge : vertexChanges.getAddedEdgeList()) { computeVertex.a
ddEdge(edge.getDestVertexId(), edge.getValue());
}
}
/**
* 3. Return the vertex object and add it to the final graph for computation.
**/
return computeVertex;
}
}
/**
* Determine actions of the vertex that participates in computation.
*
**/
public static class MyVertex extends
Vertex<LongWritable, LongWritable, LongWritable, LongWritable> {
/**
* Write the vertex edge to the result table based on the format of the input table. Ensure that the for
mat and data of the input and output tables are the same.
*
```

```
* @param context
* Indicate the runtime context.
* @param messages
* Indicate the input message.
**/
@Override
public void compute(
ComputeContext<LongWritable, LongWritable, LongWritable, LongWritable> context, Iterable<LongWri
table> messages) throws IOException {
/**
* Write the vertex ID and value to the result table that stores vertices.
**/
context.write("vertex", getId(), getValue());
/**
* Write the vertex edge to the result table that stores edges.
**/
if (hasEdges()) {
for (Edge<LongWritable, LongWritable> edge : getEdges()) { context.write("edge", getId(), edge.getDe
stVertexId(),
edge.getValue());
}
}
/**
* Perform one round of iteration.
**/
voteToHalt();
}
}
/**
* @param args
* @throws IOException
*/
public static void main(String[] args) throws IOException {
if (args.length < 4) { throw new IOException(
"Usage: VertexInputFormat <vertex input> <edge input> <vertex output> <edge output>");
}
/**
* GraphJob is used to configure Graph jobs.
*/
GraphJob job = new GraphJob();
/**
```

```
* 1. Specify input graph data and the table that stores edges.
*/
job.addInput(TableInfo.builder().tableName(args[0]).build());
job.addInput(TableInfo.builder().tableName(args[1]).build());
job.set(EDGE_TABLE, args[1]);
/**
* 2. Specify the data loading mode, and resolve the records to edges. Similar to Map, the generated ke
y is the vertex ID, and the value is the edge.
*/
job.setGraphLoaderClass(VertexInputLoader.class);
/**
* 3. Specify the data loading phase, and generate the vertex that participates in computation. Similar t
o Reduce, edges generated by Map are combined to a vertex.
*/
job.setLoadingVertexResolverClass(LoadingResolver.class);
/**
* 4. Specify actions of the vertex that participates in computation. The vertex.compute() method is use
d for each round of iteration.
*/
job.setVertexClass(MyVertex.class);
/**
* 5. Specify the output table of the Graph job, and write the computation result to the result table.
*/
job.addOutput(TableInfo.builder().tableName(args[2]).label("vertex").build());
job.addOutput(TableInfo.builder().tableName(args[3]).label("edge").build());
/**
* 6. Submit the job for execution.
*/
job.run();
}
}
```

1.10. Java SDK

MaxCompute provides Java SDK with a variety of APIs to support development on MaxCompute. For more inforamtion about the APIs, see *MaxCompute Developer Guide*.

1.11. PyODPS

1.11.1. Overview

Although you can implement most MaxCompute development by executing SQL statements, you need to use Python in complex business scenarios and custom UDF scenarios.

PyODPS is the Python SDK of MaxCompute. It provides simple and convenient Python programming interfaces, basic operations on MaxCompute objects, and the DataFrame framework. It allows you to easily analyze data on MaxCompute.

1.11.2. Quick start

This topic describes how to use a PyODPS node in DataWorks for development. The following procedure is for reference only.

1. Create a PyODPS node.
   i. Create a business flow.

      Right-click **Business Flow** below **Data Analytics** and choose **Create Workflow** from the shortcut menu.

   ii. Create a node.

      Right-click **Data Analytics** and choose **Create Data Analytics Node > PyODPS**.
2. Edit the PyODPS node.
   i. Write the program code.

      The following code is for reference only:

```
import time              # Similar to Java, module import is required before an additional SDK is
called.
import datetime          # In this example, only the print function is used.
import base64
import hashlib
import httplib
import json
import sys
import csv
from odps import ODPS      # To call an SDK related to MaxCompute, you must import this mo
dule.
def main():
   print"Hello World"      # Provide the output in the log file.
if __name__ == "__main__":   # The main entry of the program.
   main()
```

ii. Run the code.

After you edit the code, click the Run icon. You can view the running results in the **Runtime Log** section.

The first code snippet is completed. The main entry of the code is determined based on `if __name__ == "__main__"`. This statement takes effect only when the preceding script is run directly (name = main). It does not take effect when it is referenced as a module by other code files (name = Python file name).

1.11.3. Installation instructions

If you can access the Internet, we recommend that you use the Python package installer PIP to install PyODPS. For more information, see PIP installation instructions. If you want to speed up the download, we recommend that you use Alibaba Cloud images.

Prerequisites

Before you install PyOPDS, make sure that the following requirements are met:

- The Setuptools version is 3.0 or later.
- The Requests version is 2.4.0 or later.

Installation commands for reference:

```
pip install setuptools>=3.0
pip install requests>=2.4.0
```

Installation suggestions

We recommend that you install the following tools to speed up Tunnel upload:

- Greenlet. Recommended version: 0.4.10 or later.
- Cython. Recommended version: 0.19.0 or later.

Installation commands for reference:

```
pip install greenlet>=0.4.10  # Optional. It accelerates Tunnel upload.
pip install cython>=0.19.0  # Optional. It is not recommended if you use a Windows operating system.
```

> **Note** If you use a Windows operating system, make sure that you have installed Visual C++ and Cython of correct versions. Otherwise, you cannot speed up Tunnel upload.

Installation procedure

Run the following command to install PyODPS:

```
pip install pyodps
```

Run the following command to check whether the installation is complete:

```
python -c "from odps import ODPS"
```

If the Python version is not the default, you can run the following command to switch to the default version after you have installed PIP:

```
/home/tops/bin/python2.7 -m pip install setuptools>=3.0  # Replace the version of Setuptools with the actual version.
```

1.11.4. Platform instructions

1.11.4.1. Overview

PyODPS can be called as a data development node on a data development platform such as DataWorks. The platform provides a PyODPS running environment and supports scheduling and execution. You do not need to manually create a MaxCompute object. To migrate from a platform to a locally deployed PyODPS environment, see the instructions in the next topic.

1.11.4.2. Use local PyODPS

If you need to debug PyODPS locally or the resources on the platform where PyODPS is deployed cannot meet your requirements, you can deploy a local PyODPS environment.

You must install PyODPS first. For information about how to install it, see Installation instructions.

After you have installed PyODPS, you must manually create the MaxCompute object that was previously created on the platform. Then, execute the following statement on the platform to generate a statement template required by the MaxCompute object and manually modify the code:

```
print("\nfrom odps import ODPS\no = ODPS(%r, '<access-key>', %r, '<endpoint>')\n" % (o.account.access_id, o.project))
```

> **Note** You need to replace **access-key** and **endpoint** with valid values. For information about how to obtain the values, see the corresponding topics in *MaxCompute Developer Guide*.

Then, place the modified code at the beginning of all code.

1.11.4.3. Use PyODPS in DataWorks

Create a workflow node

Set Node Type to PYODPS.

MaxCompute entry

The PyODPS node in DataWorks contains a global variable odps or o, which is the MaxCompute entry. You do not need to manually define the MaxCompute entry.

```
print(o.exist_table('pyodps_iris'))
```

Execute SQL statements

For more information, see SQL.

> **Note** By default, InstanceTunnel is disabled in DataWorks, and instance.open_reader is executed by using the Result interface. In this case, a maximum of 10,000 data records can be read. After InstanceTunnel is enabled, you can execute reader.count to obtain the number of data records. If you need to obtain all data iteratively, you must disable the limit on the data volume.

You can execute the following statements to enable InstanceTunnel and disable the limit:

```
options.tunnel.use_instance_tunnel = True
options.tunnel.limit_instance_tunnel = False  # Disable the limit on the data volume.

with instance.open_reader() as reader:
    # Use InstanceTunnel to read all data.
```

You can also add `tunnel=True` and `limit=False` to open_reader to enable InstanceTunnel and disable the limit on the data volume for the current open_reader operation.

```
with instance.open_reader(tunnel=True, limit=False) as reader:
# The current open_reader operation is executed by using InstanceTunnel and all data can be read.
```

> **Note** If you do not enable InstanceTunnel, the format of the obtained data may be incorrect.

DataFrame

- Execute DataFrame.

  To execute DataFrame in DataWorks, you must explicitly call automatically executed methods, such as execute and head.

  ```
  from odps.df import DataFrame
  iris = DataFrame(o.get_table('pyodps_iris'))
  for record in iris[iris.sepal_width < 3].execute():  # Call an automatically executed method to proces
  s each record.
  ```

  To call an automatically executed method for data display, set `options.interactive` to True.

```
from odps import options
from odps.df import DataFrame
options.interactive = True  # Set options.interactive to True at the beginning of the code.
iris = DataFrame(o.get_table('pyodps_iris'))
print(iris.sepal_width.sum())  # The method is executed immediately when the system displays infor
mation.
```

- Display details.

  To display details, you must set `options.verbose`. By default, this parameter is set to True in DataWorks. The system displays details such as the LogView URL while running.

Obtain scheduling parameters.

Different from SQL nodes in DataWorks, a PyODPS node does not replace strings such as **${param_name}** in the code. Instead, it adds a dict parameter named `args` to the global variable. You can obtain the scheduling parameters in dict. This way, the Python code is not affected. For example, if you set `ds=${yyyymmdd}` under **Schedule** > **Parameter** in DataWorks, you can run the following commands to obtain the parameter value:

```
print('ds=' + args['ds'])
ds=yyyymmdd
```

> **Note** You can run the following command to obtain the partition named ds=${yyyymmdd}:
>
> ```
> o.get_table('table_name').get_partition('ds=' + args['ds'])
> ```

Limits on functions

Functions may be limited in the following aspects due to the lack of packages such as matplotlib:

- The use of the plot function of DataFrame is affected.
- User defined functions (UDFs) of DataFrame can be executed only after they are submitted to MaxCompute. As required by the Python sandbox, you can only use pure Python libraries and the NumPy library to execute UDFs. Other third-party libraries such as pandas cannot be used.
- However, you can use the NumPy and pandas libraries pre-installed in DataWorks to execute non-UDFs. You are not allowed to use other third-party libraries that contain binary code.

For compatibility reasons, `options.tunnel.use_instance_tunnel` is set to **False** in DataWorks by default. If you want to enable `InstanceTunnel` globally, you must set this parameter to **True**.

For implementation reasons, the Python atexit package is not supported. You need to use the try-finally structure to implement related functions.

Limits on usage

- The Python version of a PyODPS node is 2.7.
- Locally processed data obtained by a PyODPS node cannot exceed 50 MB, and the memory space occupied by the node cannot exceed 1 GB. Otherwise, the system stops tasks in the node. Do not write unnecessary Python data processing code in PyODPS tasks.
- Writing and debugging code in DataWorks is inefficient. We recommend that you install an IDE locally to write code.
- To prevent excess pressure on the gateway of DataWorks, the memory usage and CPU utilization are limited when PyODPS is used in DataWorks. The limits are managed by DataWorks. If the system displays **Got killed**, the memory usage exceeds the limit and the system stops the related processes. Therefore, we do not recommend you perform local data operations. However, the limits on the memory usage and CPU utilization do not apply to SQL and DataFrame tasks (except to_pandas) that are initiated by PyODPS.

1.11.5. Basic operations

1.11.5.1. Overview

PyODPS provides basic operations for MaxCompute objects. You can use Python-compliant programming methods to perform operations on MaxCompute.

1.11.5.2. Projects

You can use the `get_project` method of a MaxCompute object to obtain a project.

```
project = o.get_project('my_project')  # Obtain a specific project.
project = o.get_project()              # Obtain the default project.
```

If the my_project parameter is not specified, the default project is obtained.

You can use the `exist_project` method to check whether a project exists.

1.11.5.3. Tables

Basic operations

You can call `list_tables` for a MaxCompute object to list all tables in a project.

```
for table in o.list_tables():
# Process each table.
```

You can call `exist_table` to check whether a table exists and call `get_table` to obtain the table.

```
t = o.get_table('dual')
t.schema
odps.Schema {
  c_int_a             bigint
  c_int_b             bigint
  c_double_a          double
  c_double_b          double
  c_string_a          string
  c_string_b          string
  c_bool_a            boolean
  c_bool_b            boolean
  c_datetime_a        datetime
  c_datetime_b        datetime
}
t.lifecycle
-1
print(t.creation_time)
2014-05-15 14:58:43
t.is_virtual_view
False
t.size
1408
t.comment
'Dual Table Comment'
t.schema.columns
[<column c_int_a, type bigint>,
 <column c_int_b, type bigint>,
 <column c_double_a, type double>,
 <column c_double_b, type double>,
 <column c_string_a, type string>,
 <column c_string_b, type string>,
 <column c_bool_a, type boolean>,
 <column c_bool_b, type boolean>,
 <column c_datetime_a, type datetime>,
 <column c_datetime_b, type datetime>]
t.schema['c_int_a']
<column c_int_a, type bigint>
t.schema['c_int_a'].comment
'Comment of column c_int_a'
```

You can obtain a table from another project by specifying the project parameter.

```
t = o.get_table('dual', project='other_project')
```

Create a table schema

You can initialize a table in two ways:

- Initialize a table based on table columns and available partitions.

```
from odps.models import Schema, Column, Partition
columns = [Column(name='num', type='bigint', comment='the column'),
           Column(name='num2', type='double', comment='the column2')]
partitions = [Partition(name='pt', type='string', comment='the partition')]
schema = Schema(columns=columns, partitions=partitions)
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
schema.partitions
[<partition pt, type string>]
schema.names  # Obtain the names of non-partition fields.
['num', 'num2']
schema.types  # Obtain the types of non-partition fields.
[bigint, double]
```

- Initialize a table by calling `Schema.from_lists`. This method is easier to call, but you cannot directly set the comments of columns and partitions.

```
schema = Schema.from_lists(['num', 'num2'], ['bigint', 'double'], ['pt'], ['string'])
schema.columns
[<column num, type bigint>,
 <column num2, type double>,
 <partition pt, type string>]
```

Create a table

You can use a table schema to create a table.

```
table = o.create_table('my_new_table', schema)
table = o.create_table('my_new_table', schema, if_not_exists=True)  #The table is created only if no ta
ble with the same name exists.
table = o.create_table('my_new_table', schema, lifecycle=7)  # Set the lifecycle.
```

You can also use a comma-connected string in the "field name, field type" format to create a table. This method is easier.

```
table = o.create_table('my_new_table', 'num bigint, num2 double', if_not_exists=True)
# To create a partitioned table, you can pass in (table field list, partition field list).
table = o.create_table('my_new_table', ('num bigint, num2 double', 'pt string'), if_not_exists=True)
```

By default, when you create a table, you can only use the BIGINT, DOUBLE, DECIMAL, STRING, DATETIME, BOOLEAN, MAP, and ARRAY data types. If you need to use other data types, such as TINYINT and STRUCT, you can set `options.sql.use_odps2_extension` to True. Example:

```
from odps import options
options.sql.use_odps2_extension = True
table = o.create_table('my_new_table', 'cat smallint, content struct<title:varchar(100), body string>')
```

Synchronize table updates

After another program updates a table, such as its schema, you can call the `reload` method to synchronize the update.

```
table.reload()
```

Record

A record refers to a single row in a table. You can call new_record for the table object to create a record.

```
t = o.get_table('mytable')
r = t.new_record(['val0', 'val1'])  # The number of values must be equal to the number of fields in the ta
ble schema.
r2 = t.new_record()  # You can leave the value empty.
r2[0] = 'val0' # Set a value based on an offset.
r2['field1'] = 'val1'  # Set a value based on a field name.
r2.field1 = 'val1'  # Set a value based on an attribute.

print(record[0])  # Obtain the value at position 0.
print(record['c_double_a'])  # Obtain a value based on a field.
print(record.c_double_a)  # Obtain a value based on an attribute.
print(record[0: 3])  # Perform slicing operations.
print(record[0, 2, 3])  # Obtain values at multiple positions.
print(record['c_int_a', 'c_double_a'])  # Obtain values based on multiple fields.
```

Obtain table data

You can obtain table data in three ways:

- Call `head` to retrieve up to the first 10,000 data records in a table. Example:

```
t = odps.get_table('dual')
for record in t.head(3):
    print(record[0])  # Obtain the value at position 0.
    print(record['c_double_a'])  # Obtain a value based on a field.
    print(record[0: 3])  # Perform slicing operations.
    print(record[0, 2, 3])  # Obtain values at multiple positions.
    print(record['c_int_a', 'c_double_a'])  # Obtain values based on multiple fields.
```

- Use `open_reader` for a table object to open a reader and read the table data. You can open the reader with or without a WITH clause.

```
# Open the reader with a WITH clause:
 with t.open_reader(partition='pt=test') as reader:
    count = reader.count
    for record in reader[5:10]  # You can execute this line multiple times until all records are read. Th
e number of records is specified by count. You can change it to parallel-operation code.
        # Process a record.

# Open the reader without a WITH clause:
 reader = t.open_reader(partition='pt=test')
 count = reader.count
 for record in reader[5:10]  # You can execute this line multiple times until all records are read. The
number of records is specified by count. You can change it to parallel-operation code.
    # Process a record.
```

- Use the `read_table` method for a MaxCompute object to obtain table data. Example:

```
for record in o.read_table('test_table', partition='pt=test'):
# Process a record.
```

Write data to a table.

Similar to `open_reader`, you can use `open_writer` for a table object to open a writer and write data to a table. Example:

```
 # Open the writer with a WITH clause:
 with t.open_writer(partition='pt=test') as writer:
records = [[111, 'aaa', True],              # A list can be used.
        [222, 'bbb', False],
        [333, 'ccc', True],
        [444, 'Chinese', False]]
writer.write(records)  # records can be iterable objects.

with t.open_writer(partition='pt1=test1,pt2=test2') as writer: #Write data in multi-level partitioning m
ode.

records = [t.new_record([111, 'aaa', True]),   # Record objects can be used.
         t.new_record([222, 'bbb', False]),
         t.new_record([333, 'ccc', True]),
         t.new_record([444, 'Chinese', False])]
writer.write(records)

with t.open_writer(partition='pt=test', blocks=[0, 1]) as writer:  # Two blocks are enabled.
    writer.write(0, gen_records(block=0))
    writer.write(1, gen_records(block=1))  # The two write operations can be performed in parallel base
d on the multithreading technique. Each block is independent.

 # Open the writer without a WITH clause:
 writer = t.open_writer(partition='pt=test', blocks=[0, 1])
 writer.write(0, gen_records(block=0))
 writer.write(1, gen_records(block=1))
 writer.close()  # You must close the writer. Otherwise, the written data may be incomplete.
```

If the specified partition does not exist, use the create_partition parameter to create a partition. Example:

```
with t.open_writer(partition='pt=test', create_partition=True) as writer:
   records = [[111, 'aaa', True],              # A list can be used.
           [222, 'bbb', False],
           [333, 'ccc', True],
           [444, 'Chinese', False]]
   writer.write(records)  # records can be iterable objects.
```

You can also use the `write_table` method for a MaxCompute object to write data. Example:

```
records = [[111, 'aaa', True],          # A list can be used.
           [222, 'bbb', False],
           [333, 'ccc', True],
           [444, 'Chinese', False]]
o.write_table('test_table', records, partition='pt=test', create_partition=True)
```

> **Note**
>
> - Each time you call write_table, MaxCompute generates a file on the server. This operation is time-consuming. If too many files are generated, the efficiency of subsequent queries will be reduced. Therefore, we recommend that you write multiple records at a time or provide a Generator object when you use the write_table method.
> - When you use `write_table`, new data will be appended to existing data. PyODPS does not provide options to overwrite existing data. You need to manually remove the data that you want to overwrite. For a non-partitioned table, you must call `table.truncate()`. For a partitioned table, you need to delete partitions first.

Delete a table

```
o.delete_table('my_table_name', if_exists=True)  # Delete a table if the table exists.
t.drop()  # The drop function can be directly executed if a table object exists.
```

Create a DataFrame object

PyODPS provides a DataFrame framework, which allows you to conveniently query and manage MaxCompute data. You can use `to_df` to convert a table to a DataFrame object.

```
table = o.get_table('my_table_name')
df = table.to_df()
```

Table partitions

- Basic operations

  Check whether a table is partitioned:

  ```
  if table.schema.partitions:
      print('Table %s is partitioned.' % table.name)
  ```

  Iterate over all the partitions in a table:

```
for partition in table.partitions:
    print(partition.name)
for partition in table.iterate_partitions(spec='pt=test'):
# Iterate over subpartitions.
```

Check whether a specific partition exists:

```
table.exist_partition('pt=test,sub=2019')
```

Obtain a specific partition:

```
partition = table.get_partition('pt=test')
 print(partition.creation_time)
2019-09-18 22:22:27
 partition.size
0
```

- Create a partition.

```
t.create_partition('pt=test', if_not_exists=True)  # Create the partition if it does not exist.
```

- Delete a partition.

```
t.delete_partition('pt=test', if_exists=True)  # Delete the partition if it exists.
partition.drop()  # Use the drop method to delete the partition object if it exists.
```

Data upload and download channels

MaxCompute Tunnel is the data channel of MaxCompute. You can use it to upload data to or download data from MaxCompute.

> **Note** We recommend that you use the write and read interfaces of tables instead of the Tunnel interface. In a Cython environment, PyODPS compiles C programs during installation to accelerate the Tunnel upload and download.

- Upload example:

```
from odps.tunnel import TableTunnel

table = o.get_table('my_table')

tunnel = TableTunnel(odps)
upload_session = tunnel.create_upload_session(table.name, partition_spec='pt=test')

with upload_session.open_record_writer(0) as writer:
    record = table.new_record()
    record[0] = 'test1'
    record[1] = 'id1'
    writer.write(record)

    record = table.new_record(['test2', 'id2'])
    writer.write(record)

upload_session.commit([0])
```

- Download example:

```
from odps.tunnel import TableTunnel

tunnel = TableTunnel(odps)
download_session = tunnel.create_download_session('my_table', partition_spec='pt=test')

with download_session.open_record_reader(0, download_session.count) as reader:
    for record in reader:
# Process each record.
```

> **Note** PyODPS currently does not allow you to upload external tables.

1.11.5.4. SQL

PyODPS supports MaxCompute SQL queries and provides methods to read execution results.

You can use the `execute_sql` and `run_sql` methods to create task instances.

> **Note** The commands that are executable in the MaxCompute console may not be executed as SQL statements in MaxCompute. Use other methods to execute non-DDL/DML statements. For example, use the `run_security_query` method to execute GRANT/REVOKE statements, and use the `run_xflow` or `execute_xflow` method to execute PAI commands.

Execute SQL statements

```
o.execute_sql('select * from dual')  # Run the SQL statement in synchronous mode. Blocking continues until execution of the SQL statement is completed.
instance = o.run_sql('select * from dual')  # Run the SQL statement in asynchronous mode.
print(instance.get_logview_address())  # Obtain the LogView address.
instance.wait_for_success()  # Blocking continues until execution of the SQL statement is completed.
```

Configure runtime parameters

You can configure runtime parameters by setting the `hints` parameter. The type of this parameter is DICT.

```
o.execute_sql('select * from pyodps_iris', hints={'odps.sql.mapper.split.size': 16})
```

If you set the `sql.settings` parameter globally, you need to configure runtime parameters each time you execute the statement.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from pyodps_iris')  # The hints parameter is automatically set based on global s ettings.
```

Obtain SQL query results

You can directly call the `open_reader` method to obtain SQL query results. In the following example, structured data is returned.

```
with o.execute_sql('select * from dual').open_reader() as reader:
    for record in reader:
    # Process each record.
```

When the `DESC` command is executed, you can use `reader.raw` to obtain the raw SQL query results.

```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

If `options.tunnel.use_instance_tunnel` is set to True when you use `open_reader`, PyODPS calls Instance Tunnel by default. Otherwise, PyODPS calls the old Result interface. However, if you are using an earlier version of MaxCompute or an error occurs when PyODPS calls Instance Tunnel, PyODPS generates an alert and automatically downgrades the call object to the old Result interface. You can identify the cause of the downgrade based on the alert information. If the result of Instance Tunnel does not meet your expectation, set this option to False. When you call `open_reader`, you can specify a Result interface by setting the tunnel parameter.

```
# Call Instance Tunnel.
with o.execute_sql('select * from dual').open_reader(tunnel=True) as reader:
    for record in reader:
    # Process each record.
# Call the Result interface.
with o.execute_sql('select * from dual').open_reader(tunnel=False) as reader:
    for record in reader:
    # Process each record.
```

By default, PyODPS does not limit the amount of data that can be read from an instance. For protected projects, the amount of data that can be downloaded by using Tunnel is limited. If `options.tunnel.limit_instance_tunnel` is not set, the limit is automatically enabled. The number of data entries that can be downloaded is limited based on project configurations. Generally, a maximum of 10,000 data entries can be downloaded. You can add a limit option to the `open_reader` method or set `options.tunnel.limit_instance_tunnel` to True to manually limit the amount.

If your MaxCompute version only supports the old Result interface and you need to read all data, you can export the SQL query results to another table and then use the table read interface to read data. This may be limited by project security settings.

In PyODPS V0.7.7.1 and later, you can use `open_reader` to call Instance Tunnel to obtain all data.

```
instance = o.execute_sql('select * from movielens_ratings limit 20000')
with instance.open_reader() as reader:
    print(reader.count)
     # for record in reader: Traverse the 20,000 data records. In this example, only 10 data records are o
btained based on data slicing.
    for record in reader[:10]:
        print(record)
```

Set alias

During runtime, if the resource referenced by a UDF changes dynamically, you can use the old resource name as the alias of the current resource. This way, you do not need to delete the UDF or create a UDF.

```
from odps.models import Schema
myfunc = '''\
from odps.udf import annotate
from odps.distcache import get_cache_file

@annotate('bigint->bigint')
class Example(object):
    def __init__(self):
        self.n = int(get_cache_file('test_alias_res1').read())

    def evaluate(self, arg):
        return arg + self.n
'''
res1 = o.create_resource('test_alias_res1', 'file', file_obj='1')
o.create_resource('test_alias.py', 'py', file_obj=myfunc)
o.create_function('test_alias_func',
                  class_type='test_alias.Example',
                  resources=['test_alias.py', 'test_alias_res1'])

table = o.create_table(
    'test_table',
    schema=Schema.from_lists(['size'], ['bigint']),
    if_not_exists=True
)

data = [[1, ], ]
# Write a row of data that contains only one value: 1.
o.write_table(table, 0, [table.new_record(it) for it in data])

with o.execute_sql(
    'select test_alias_func(size) from test_table').open_reader() as reader:
    print(reader[0][0])
res2 = o.create_resource('test_alias_res2', 'file', file_obj='2')
# Use the name of the resource whose content is 1 as the alias of the resource whose content is 2. Yo
u do not need to modify the UDF or resource.
with o.execute_sql(
    'select test_alias_func(size) from test_table',
    aliases={'test_alias_res1': 'test_alias_res2'}).open_reader() as reader:
    print(reader[0][0])
```

Execute SQL statements in an interactive environment

In IPython and Jupyter, SQL plug-ins can be used to execute SQL statements and parameterized queries are supported.

Set biz_id

Occasionally, when an SQL statement is executed, biz_id must be submitted. Otherwise, an error occurs. In this case, you can set biz_id in the global options to troubleshoot the error.

```
from odps import options
options.biz_id = 'my_biz_id'
o.execute_sql('select * from pyodps_iris')
```

1.11.5.5. Task instances

Tasks, such as SQL tasks, are the basic computing unit of MaxCompute.

Basic operations

Tasks are implemented as MaxCompute instances. You can call `list_instances` to obtain all instances in a project, `exist_instance` to check whether an instance exists, and `get_instance` to obtain an instance.

```
for instance in o.list_instances():
    print(instance.id)
    o.exist_instance('my_instance_id')
```

You can call the `stop` method for an instance or call `stop_instance` at the MaxCompute entry to stop an instance.

Obtain the LogView address of a task

You can call `get_logview_address` to obtain the LogView address of an SQL task.

```
# Obtain the LogView address based on an existing instance object.
instance = o.run_sql('desc pyodps_iris')
print(instance.get_logview_address())
# Obtain the LogView address based on an instance ID.
instance = o.get_instance('20160426055209459g9k5pvyi2')
print(instance.get_logview_address())
```

For an XFlow task, you must enumerate its subtasks and then obtain the LogView address of each subtask.

```
instance = o.run_xflow('AppendID', 'algo_public', {'inputTableName': 'input_table', 'outputTableName': 'o
utput_table'})
for sub_inst_name, sub_inst in o.get_xflow_sub_instances(instance).items():
    print('%s: %s' % (sub_inst_name, sub_inst.get_logview_address()))
```

Obtain the status of a task instance

An instance can be in the **Running**, **Suspended**, or **Terminated** state. You can obtain the status from the status attribute. You can call the `is_terminated` method to check whether the execution of the current instance is completed and call the `is_successful` method to check whether the execution is successful. If the task is running or fails, False is returned for both methods.

```
instance = o.get_instance('2016042605520945g9k5pvyi2')
instance.status
<Status.TERMINATED: 'Terminated'>
    from odps.models import Instance
    instance.status == Instance.Status.TERMINATED
    True
    instance.status.value
    'Terminated'
```

You can call the `wait_for_completion` method to ask the system to return the status until the instance execution is completed. The `wait_for_success` method functions similarly. The difference is that if the execution fails, an exception is reported.

Subtask operations

A running instance may contain one or more subtasks, which are called Tasks. Note that Task here does not refer to the basic computing unit of MaxCompute.

You can call the `get_task_names` method to obtain the names of all Tasks.

```
instance.get_task_names()
['SQLDropTableTask']
```

After obtaining a Task name, you can call `get_task_result` to obtain the execution result of the Task. You can also call `get_task_results` to obtain the execution result of each Task in the form of a dictionary.

```
instance = o.execute_sql('select * from pyodps_iris limit 1')
instance.get_task_names()
['AnonymousSQLTask']
instance.get_task_result('AnonymousSQLTask')
'"sepallength","sepalwidth","petallength","petalwidth","name"\n5.1,3.5,1.4,0.2,"Iris-setosa"\n'
instance.get_task_results()
OrderedDict([('AnonymousSQLTask', "sepallength","sepalwidth","petallength","petalwidth","name"\n5.
1,3.5,1.4,0.2,"Iris-setosa"\n')])
```

You can call `get_task_progress` to obtain the current running progress of a Task when the task instance is running.

```
while not instance.is_terminated():
    for task_name in instance.get_task_names():
        print(instance.id, instance.get_task_progress(task_name).get_stage_progress_formatted_string()
)
        time.sleep(10)
20190519101349613gzbzufck2 2019-05-19 18:14:03 M1_Stg1_job0:0/1/1[100%]
```

1.11.5.6. Resources

PyODPS mainly supports file and table resources.

Resources commonly apply to UDFs and MapReduce on MaxCompute.

You can call `list_resources` to list all resources, `exist_resource` to check whether a resource exists, and `delete_resource` to delete a resource. You can also call the `drop` method to delete a resource.

File resources

Files of basic file types and the .py, .jar, and .archive types are supported.

- Create a file resource

  You can create a file resource by specifying a resource name, a file type, and a file-like or string object. Example:

  ```
  resource = odps.create_resource('test_file_resource', 'file', file_obj=open('/to/path/file')) # Use a fil
  e-like object.
  resource = odps.create_resource('test_py_resource', 'py', file_obj='import this') # Use a string object
  .
  ```

- Read and modify a file resource

You can call the `open` method for a file resource or call `open_resource` at the MaxCompute entry to open a file resource. The opened object is a file-like object. Similar to the `open` method built in Python, file resources also support various opening modes. Example:

```
with resource.open('r') as fp:  # Open the file in read mode.
  content = fp.read()  # Read all content.
  fp.seek(0)  # Return to the beginning of the resource.
  lines = fp.readlines()  # Read multiple lines.
  fp.write('Hello World') # Error. Data cannot be written into a file in read mode.

with odps.open_resource('test_file_resource', mode='r+') as fp:  # Open the file in read/write mode.
  fp.read()
  fp.tell()  # Current position
  fp.seek(10)
  fp.truncate()  # Truncate the file to the specified length.
  fp.writelines(['Hello\n', 'World\n'])  # Write multiple lines into the file.
  fp.write('Hello World')
  fp.flush()  # Manually calling the method will submit the update to MaxCompute.
```

PyODPS supports the following opening modes:

- r: read mode. The file can be opened but data cannot be written into it.
- w: write mode. Data can be written into the file, but data in it cannot be read. Note that the file content is cleared first if the file is opened in write mode.
- a: append mode. Data can be added to the end of the file.
- r+: read/write mode. You can read and write data from and to the file.
- w+: This mode is similar to r+, but the file content is cleared first.
- a+: This mode is similar to r+, but data can only be written to the end of the file.

In PyODPS, file resources can be opened in binary mode. For example, some compressed files must be opened in binary mode. **rb** indicates that a file is opened in binary read mode, and **r+b** indicates that a file is opened in binary read/write mode.

Table resources

- Create a table resource

```
o.create_resource('test_table_resource', 'table', table_name='my_table', partition='pt=test')
```

- Update a table resource

```
table_resource = o.get_resource('test_table_resource')
table_resource.update(partition='pt=test2', project_name='my_project2')
```

- Obtain a table and a partition

```
table_resource = o.get_resource('test_table_resource')
table = table_resource.table
print(table.name)
partition = table_resource.partition
print(partition.spec)
```

- Read and write data

```
table_resource = o.get_resource('test_table_resource')
with table_resource.open_writer() as writer:
    writer.write([0, 'aaaa'])
    writer.write([1, 'bbbbb'])
    with table_resource.open_reader() as reader:
        for rec in reader:
            print(rec)
```

1.11.5.7. Functions

You can create UDFs and use them in MaxCompute SQL.

Basic operations

You can call `list_functions` of a MaxCompute object to obtain all functions in the project. You can call `exist_function` to check whether a function exists and call `get_function` to obtain a function.

Create a function

```
# Reference resources in the current project.
resource = o.get_resource('my_udf.py')
function = o.create_function('test_function', class_type='my_udf.Test', resources=[resource])
# Reference resources in another project.
resource2 = o.get_resource('my_udf.py', project='another_project')
function2 = o.create_function('test_function2', class_type='my_udf.Test', resources=[resource2])
```

Delete a function

```
o.delete_function('test_function')
function.drop()  # Call the drop method if the function exists.
```

Update a function

You can call the `update` method to update a function.

```
function = o.get_function('test_function')
new_resource = o.get_resource('my_udf2.py')
function.class_type = 'my_udf2.Test'
function.resources = [new_resource, ]
function.update()  # Update the function.
```

1.11.6. DataFrame

PyODPS provides an interface similar to pandas called PyODPS DataFrame. This interface operates on MaxCompute tables and makes full use of the capabilities of MaxCompute. You can also change the data source from MaxCompute tables to pandas DataFrame, so that the same code can be executed on pandas.

This topic describes how to create and perform operations on DataFrame objects and how to use DataFrame to process data. For the complete DataFrame document, see DataFrame.

DataFrame object operations

The following example demonstrates how to create a DataFrame object. It is for reference only.

> **Note** The data used in the example is downloaded from movielens 100K. You can download the data as needed.

Assume that the following three tables already exist: pyodps_ml_100k_movies (movie-related data), pyodps_ml_100k_users (user-related data), and pyodps_ml_100k_ratings (rating-related data).

1. If no MaxCompute object is provided in the runtime environment, you must create an object.

   ```
   from odps import ODPS
   o = ODPS('**your-access-id**', '**your-secret-access-key**',
           project='**your-project**', endpoint='**your-end-point**')
   ```

2. You only need to import a Table object to create a DataFrame object.

   ```
   from odps.df import DataFrame
   users = DataFrame(o.get_table('pyodps_ml_100k_users'))
   ```

3. You can use the dtypes attribute to view the fields of DataFrame and the types of these fields.

```
users.dtypes
odps.Schema {
 user_id          int64
 age              int64
 sex              string
 occupation       string
 zip_code         string
}
```

4. You can use the `head` method to obtain the first N data records for quick data preview. Example:

```
users.head(10)
   user_id  age  sex    occupation  zip_code
0        1   24    M    technician     85711
1        2   53    F         other     94043
2        3   23    M        writer     32067
3        4   24    M    technician     43537
4        5   33    F         other     15213
5        6   42    M     executive     98101
6        7   57    M  administrator     91344
7        8   36    M  administrator     05201
8        9   29    M       student     01002
9       10   53    M        lawyer     90703
```

5. You can add a filter on the fields if you do not want to view all of them.
   - Example:

```
users[['user_id', 'age']].head(5)
   user_id  age
0        1   24
1        2   53
2        3   23
3        4   24
4        5   33
```

   - You can also exclude several fields. Example:

```
users.exclude('zip_code', 'age').head(5)
  user_id  sex  occupation
0       1    M  technician
1       2    F       other
2       3    M      writer
3       4    M  technician
4       5    F       other
```

- When you exclude some fields, you may want to obtain new columns through computation. For example, add the sex_bool attribute and set it to True if sex is M. Otherwise, set it to False.

```
users.select(users.exclude('zip_code', 'sex'), sex_bool=users.sex == 'M').head(5)
  user_id  age  occupation  sex_bool
0       1   24  technician      True
1       2   53       other     False
2       3   23      writer      True
3       4   24  technician      True
4       5   33       other     False
```

6. Obtain the numbers of male and female users.

```
users.groupby(users.sex).agg(count=users.count())
  sex  count
0   F    273
1   M    670
```

7. To divide users by occupation, obtain the first 10 occupations that have the largest population, and sort the occupations in descending order of population.

```
df = users.groupby('occupation').agg(count=users['occupation'].count())
df.sort(df['count'], ascending=False)[:10]
     occupation  count
0       student    196
1         other    105
2      educator     95
3  administrator     79
4      engineer     67
5    programmer     66
6     librarian     51
7        writer     45
8     executive     32
9     scientist     31
```

Alternatively, you can use the `value_counts` method. Note that the number of records returned by this method is limited by `options.df.odps.sort.limit`.

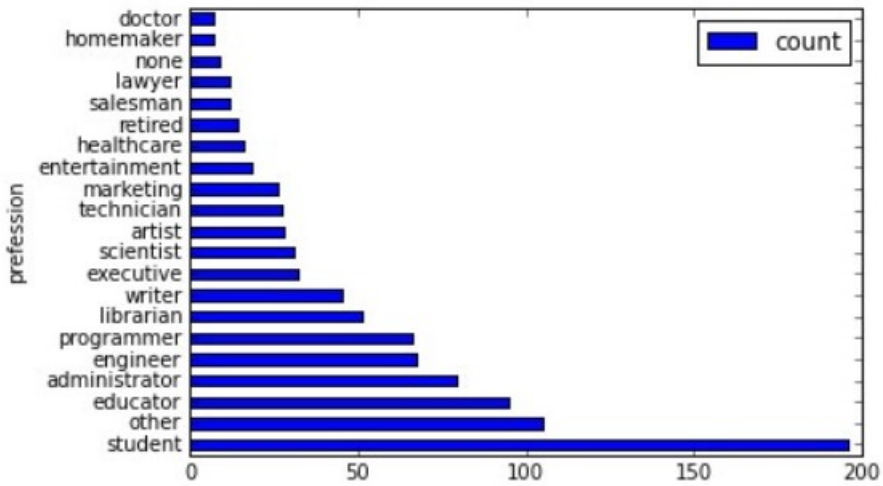
```
users.occupation.value_counts()[:10]
    occupation  count
0      student    196
1        other    105
2     educator     95
3 administrator     79
4     engineer     67
5   programmer     66
6    librarian     51
7       writer     45
8    executive     32
9    scientist     31
```

8. Show data in a more intuitive graph.

```
%matplotlib inline
```

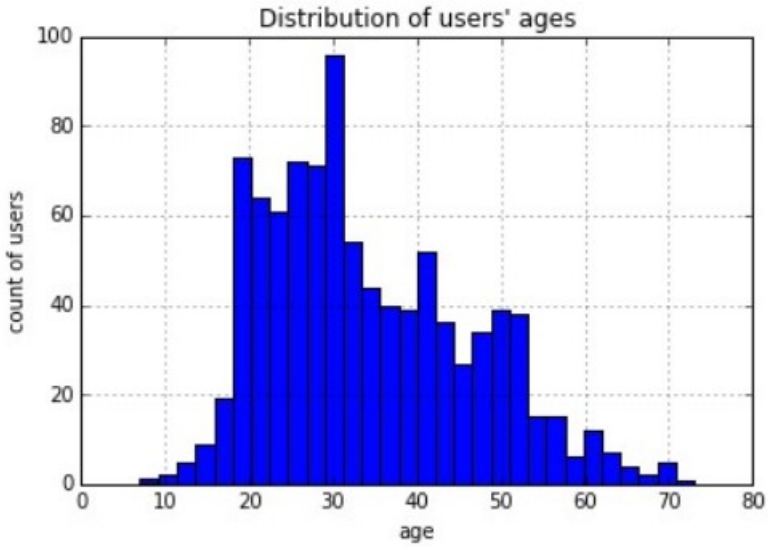
9. Use a horizontal column chart to visualize data.

```
users['occupation'].value_counts().plot(kind='barh', x='occupation', ylabel='prefession')
<matplotlib.axes._subplots.AxesSubplot at 0x10653cfd0>
```



10. Divide users into 30 groups by age and view the histogram of age distribution.

```
users.age.hist(bins=30, title="Distribution of users' ages", xlabel='age', ylabel='count of users')
<matplotlib.axes._subplots.AxesSubplot at 0x10667a510>
```



11. Use `join` to join the three tables and save them as a new table.

```
movies = DataFrame(o.get_table('pyodps_ml_100k_movies'))
ratings = DataFrame(o.get_table('pyodps_ml_100k_ratings'))

o.delete_table('pyodps_ml_100k_lens', if_exists=True)
lens = movies.join(ratings).join(users).persist('pyodps_ml_100k_lens')

lens.dtypes
odps.Schema {
  movie_id                int64
  title                   string
  release_date            string
  video_release_date      string
  imdb_url                string
  user_id                 int64
  rating                  int64
  unix_timestamp          int64
  age                     int64
  sex                     string
  occupation              string
  zip_code                string
}
```

12. Divide users aged 0 to 80 into eight age groups.

```
labels = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
cut_lens = lens[lens, lens.age.cut(range(0, 81, 10), right=False, labels=labels).rename('age_group'
)]
```

13. View the first 10 data records of a single age in a group.

```
cut_lens['age_group', 'age'].distinct()[:10]
  age_group  age
0     0-9    7
1    10-19   10
2    10-19   11
3    10-19   13
4    10-19   14
5    10-19   15
6    10-19   16
7    10-19   17
8    10-19   18
9    10-19   19
```

14. View the total rating and average rating of users in each age group.

```
cut_lens.groupby('age_group').agg(cut_lens.rating.count().rename('total_rating'), cut_lens.rating.
mean().rename('avg_rating'))
    age_group  avg_rating  total_rating
0     0-9  3.767442       43
1    10-19  3.486126     8181
2    20-29  3.467333    39535
3    30-39  3.554444    25696
4    40-49  3.591772    15021
5    50-59  3.635800     8704
6    60-69  3.648875     2623
7    70-79  3.649746      197
```

DataFrame data processing

The following example demonstrates how to process DataFrame data. It is for reference only.

> **Note** The data used in the example is downloaded from Iris dataset. You can download the data as needed.

1. Create a test data table.

   Use the table management feature of DataWorks to create a table, and then click **DDL mode**.

   Enter the CREATE TABLE statement and submit the table. Example:

```
CREATE TABLE `pyodps_iris` (
    `sepallength` double COMMENT 'sepallength(cm)',
    `sepalwidth` double COMMENT 'sepalwidth(cm)',
    `petallength` double COMMENT 'petallength(cm)',
    `petalwidth` double COMMENT 'petalwidth(cm)',
    `name` string COMMENT 'name'
) ;
```

2. Upload test data.

   Click the **Import** icon.

   Enter the table name and upload the downloaded dataset.

   Select **By Location** and click Import.

3. Create a PyODPS node to store and run code.
4. Enter the code and click the Run icon. You can view the result in the **Runtime Log** section in the lower pane.

   Code details:

```
from odps.df import DataFrame
from odps.df import output
iris = DataFrame(o.get_table('pyodps_iris')) #Create the DataFrame object iris from the MaxCompu
te table.
print iris.head(10)
print iris.sepallength.head(5) # Display part of the iris content.

# Use a user defined function to calculate the sum of two columns of iris.
print iris.apply(lambda row: row.sepallength + row.sepalwidth, axis=1, reduce=True, types='float')
.rename('sepaladd').head(3)

# Specify the output name and type of the function.
@output(['iris_add', 'iris_sub'], ['float', 'float'])
def handle(row):
# Use the yield keyword to return multiple rows of results.
    yield row.sepallength - row.sepalwidth,row.sepallength + row.sepalwidth
    yield row.petallength - row.petalwidth,row.petallength + row.petalwidth
# Display the results of the first five rows. axis=1 indicates that the axis of the column extends h
orizontally.
print iris.apply(handle,axis=1).head(5)
```

1.11.7. User experience enhancement

1.11.7.1. Command line

PyODPS provides an enhanced command line tool.

You can perform the following steps to configure and call the tool:

1. Import the PyODPS enhancement tool:

```
from odps.inter import setup, enter, teardown
```

2. Configure your account:

```
setup('**your-access_id**', '**your-access-key**', '**your-project**', endpoint='**your-endpoint**')
```

> **Note**
> - If you do not specify the room parameter, the default room is used.
> - After you have configured an account, you do not need to enter the account information again upon the next logon.

3. You can then call the enter method to create a Room object in any Python interactive interface.

```
room = enter()
o = room.odps
o.get_table('dual')
odps.Table
 name: odps_test_sqltask_finance.`dual`
 schema:
  c_int_a             : bigint
  c_int_b             : bigint
  c_double_a          : double
  c_double_b          : double
  c_string_a          : string
  c_string_b          : string
  c_bool_a            : boolean
  c_bool_b            : boolean
  c_datetime_a        : datetime
  c_datetime_b        : datetime
```

> **Note** The MaxCompute object is not automatically updated when you change the setup of the room. You need to call `enter()` again to retrieve the new Room object.

After you have configured an account and called objects, you can store, retrieve, or delete objects in the room or delete the entire room.

- You can store commonly used MaxCompute tables or resources in the room. Example:

```
room.store('stored-table', o.get_table('dual'), desc='Simple stored table example')
```

- You can call the `display` method to display the stored objects as a table. Example:

```
room.display()
```

default name	desc
stored-table	Simple stored table example
iris	Iris dataset

- You can use `room['stored-table']` or `room.iris` to retrieve the stored objects. Example:

```
room['stored-table']
odps.Table
 name: odps_test_sqltask_finance.`dual`
 schema:
   c_int_a             : bigint
   c_int_b             : bigint
   c_double_a          : double
   c_double_b          : double
   c_string_a          : string
   c_string_b          : string
   c_bool_a            : boolean
   c_bool_b            : boolean
   c_datetime_a        : datetime
   c_datetime_b        : datetime
```

- You can call the `drop` method to delete objects in the room.

```
room.drop('stored-table')
room.display()
default name       desc
iris       Iris dataset
```

- You can call `teardown()` to delete a room. When no parameter is specified, the default room is deleted.

```
teardown()
```

1.11.7.2. IPython

PyODPS provides the IPython plug-in to facilitate MaxCompute operations.

IPython enhancement

Some commands are provided for command line enhancement.

- You can run the following code to load the plug-in:

```
%load_ext odps
%enter
```

Result:

```
<odps.inter.Room at 0x11341df10>
```

- In this case, the global `o` and `odps` variables can be retrieved. You can run the `o.get_table` or `odps.get_table` command to call a table.

```
o.get_table('dual')
odps.get_table('dual')
```

Result:

```
odps.Table
  name: odps_test_sqltask_finance.`dual`
  schema:
    c_int_a             : bigint
    c_int_b             : bigint
    c_double_a          : double
    c_double_b          : double
    c_string_a          : string
    c_string_b          : string
    c_bool_a            : boolean
    c_bool_b            : boolean
    c_datetime_a        : datetime
    c_datetime_b        : datetime
```

- Run the following command to display the stored objects as a table:

```
%stores
```

Result:

default name	desc
iris	Iris dataset

Object name completion

PyODPS enhances the code completion feature that is provided by IPython. When you write statements such as o.get_xxx, the object name is automatically completed. In the following examples, <tab> is used to denote pressing the Tab key. When you enter the statement and encounter <tab>, press the Tab key.

- Use the Tab key to complete the object name:

```
o.get_table(<tab>
```

- You can enter the first few characters of an object name and press the Tab key to complete it:

```
o.get_table('tabl<tab>
```

  IPython auto-completes the table name that starts with tabl.

- This feature can also complete the names of objects in different projects. The syntax is as follows:

```
o.get_table(project='project_name', name='tabl<tab>
o.get_table('tabl<tab>', project='project_name')
```

- If multiple matching objects exist, IPython provides a list. `options.completion_size` specifies the maximum number of objects in the list. The default value is 10.

SQL statements

PyODPS provides a SQL plug-in to execute MaxCompute SQL statements.

- You can use `%sql` to execute a single-line SQL statement. Example:

```
In [*]: %sql select * from pyodps_iris limit 5
|==========================================|  1 /  1  (100.00%)       3s
Out[*]:
   sepallength  sepalwidth  petallength  petalwidth       name
0          5.1         3.5          1.4         0.2  Iris-setosa
1          4.9         3.0          1.4         0.2  Iris-setosa
2          4.7         3.2          1.3         0.2  Iris-setosa
3          4.6         3.1          1.5         0.2  Iris-setosa
4          5.0         3.6          1.4         0.2  Iris-setosa
```

- You can use `%%sql` to execute a multiple-line SQL statement. Example:

```
In [*]: %%sql
  ....: select * from pyodps_iris
  ....: where sepallength < 5
  ....: limit 5
  ....:
|==========================================|  1 /  1  (100.00%)       15s
Out[*]:
  sepallength  sepalwidth  petallength  petalwidth       name
0         4.9         3.0          1.4         0.2  Iris-setosa
1         4.7         3.2          1.3         0.2  Iris-setosa
2         4.6         3.1          1.5         0.2  Iris-setosa
3         4.6         3.4          1.4         0.3  Iris-setosa
4         4.4         2.9          1.4         0.2  Iris-setosa
```

- To execute parameterized SQL statements, you can use `:parameter` to specify the parameter. Example:

```
In [1]: %load_ext odps

In [2]: mytable = 'dual'

In [3]: %sql select * from :mytable
|==========================================|  1 /  1  (100.00%)       2s
Out[3]:
  c_int_a  c_int_b  c_double_a  c_double_b  c_string_a  c_string_b c_bool_a  \
0       0       0       -1203           0           0       -1203     True

  c_bool_b       c_datetime_a       c_datetime_b
0    False  2012-03-30 23:59:58  2012-03-30 23:59:59
```

- For SQL runtime parameters, you can use `%set` to set a global parameter or use SET within a SQLCell to set a local parameter. The following example sets a local parameter, which does not affect the global setting.

```
In [*]: %%sql
        set odps.sql.mapper.split.size = 16;
        select * from pyodps_iris;
```

- The following example sets a global parameter. Global settings apply to all subsequent SQL statements.

```
In [*]: %set odps.sql.mapper.split.size = 16
```

Upload pandas DataFrame to MaxCompute tables

PyODPS provides the following command to upload pandas DataFrame objects to MaxCompute tables:

You only need to use the `%persist` command. The first parameter df is the variable name. The second parameterpyodps_pandas_df is the MaxCompute table name.

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.arange(9).reshape(3, 3), columns=list('abc'))
%persist df pyodps_pandas_df
```

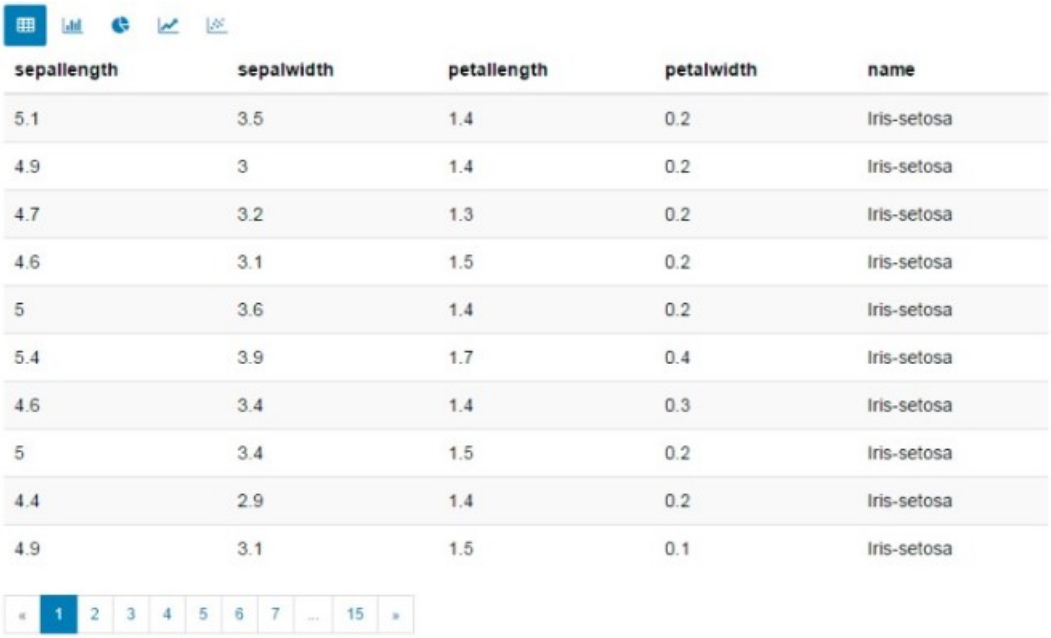
1.11.7.3. Jupyter Notebook

PyODPS enhances the result exploration and progress display features of Jupyter Notebook.

Result exploration

PyODPS provides a data exploration feature in Jupyter Notebook for SQLCell and DataFrame. You can use interactive data exploration tools to browse local data and create graphs.

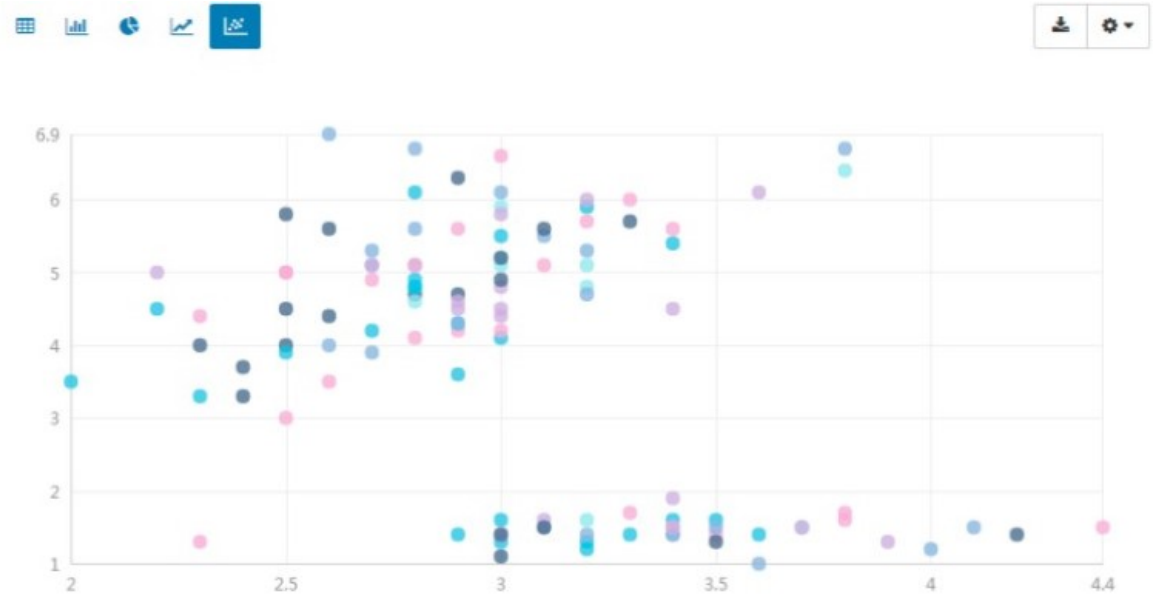
1. If the execution result is a DataFrame object, PyODPS reads the result and displays it in a paged table. You can click a page number or the Previous or Next button to browse the data.



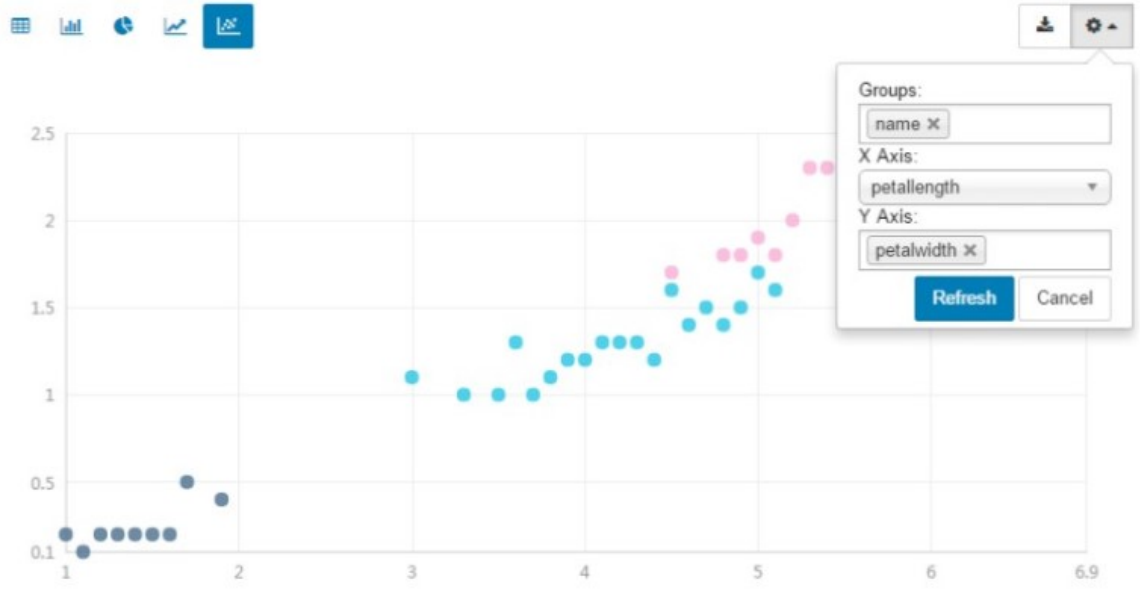
sepallength	sepalwidth	petallength	petalwidth	name
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa

« 1 2 3 4 5 6 7 ... 15 »

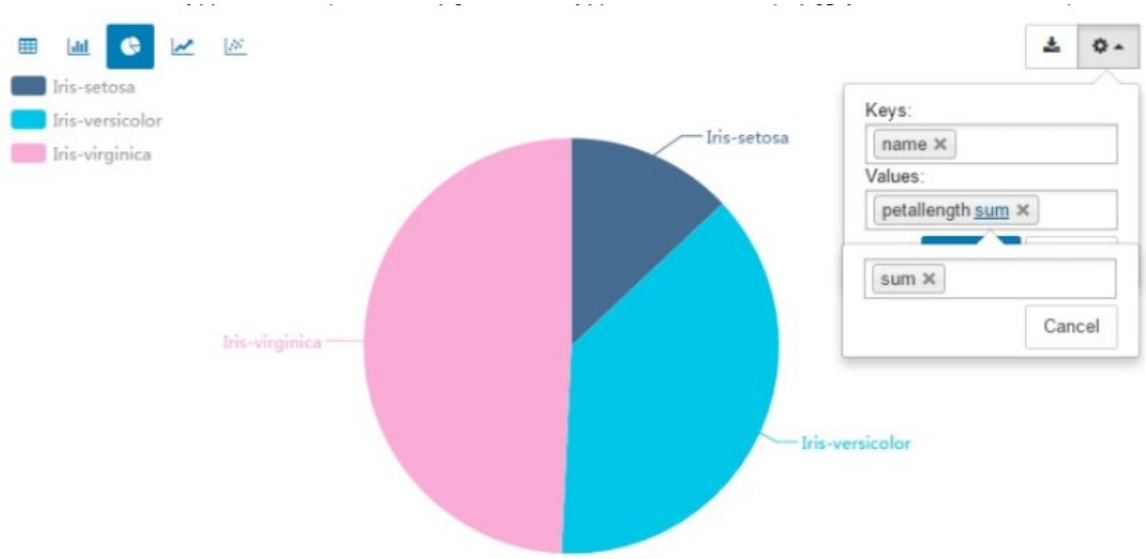
2. You can select other display modes on the top of the table to display the result in a column chart, pie chart, line chart, or scatter chart. The following figure shows a scatter chart created based on the default fields, which are the first three fields.



3. You can click the settings icon in the upper-right corner of a graph to modify the settings. For example, set Groups to **name**, X Axis to **petallength**, and Y Axis to **petalwidth**. The result is shown in the following graph. The **petallength-petalwidth** setting displays the data in a manner that is easy to understand.



For bar charts and pie charts, you can select an aggregate function for the value fields. The default aggregate function for column charts is `sum`, and that for pie charts is `count`. You can click the function name next to the value field name to select another function. For line charts, ensure that the values on the x-axis are not null. Otherwise, the graph may not display correctly.



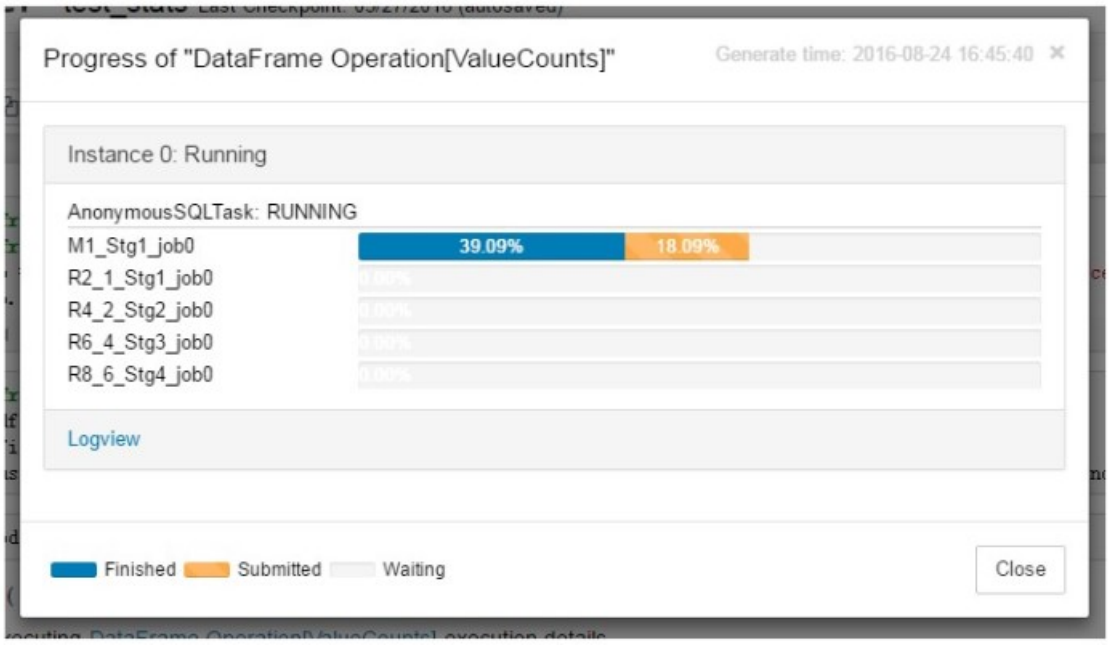
4. After you finish drawing the graph, click the **Download** icon to save it.

> **Note** Before you use this feature, ensure that you have installed pandas and ipywidgets.

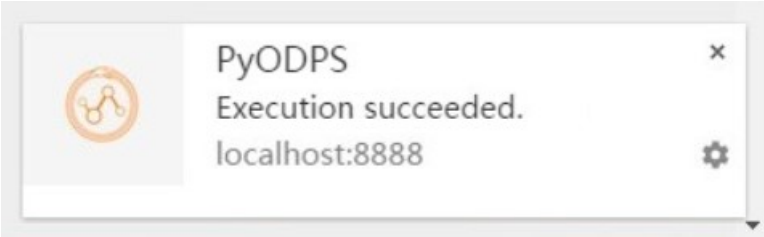
Progress display

The execution of large jobs takes extended periods of time. PyODPS provides progress bars to show the execution progress. When DataFrame jobs, machine learning jobs, or SQL statements that start with `%sql` are executed in Jupyter Notebook, a list of these jobs and their overall progress are displayed.

When you click a job name, a dialog box that displays the progress of each task in the job appears.



After the execution has been completed, a message that indicates whether the job is successful appears.



1.11.8. Configuration

PyODPS provides a series of configuration options. You can obtain them by using `odps.options`. Example:

```
from odps import options
# Set the lifecycle option, which specifies the lifecycle of all output tables.
options.lifecycle = 30
# Set the tunnel.string_as_binary option to True to use bytes instead of unicode to download data of t
he STRING type.
options.tunnel.string_as_binary = True
# When you execute PyODPS DataFrame on MaxCompute, you can refer to the following configuration
to set the limit to a relatively large value during a sort operation.
options.df.odps.sort.limit = 100000000
```

General configuration

Option	Description	Default value
end_point	The endpoint of MaxCompute.	None
default_project	The default project.	None
log_view_host	The hostname of LogView.	None
log_view_hours	The retention time of LogView (unit: hours).	24
local_timezone	The time zone used. True indicates local time, and False indicates UTC. The time zone of pytz can also be used.	None
lifecycle	The lifecycle of all tables.	None
temp_lifecycle	The lifecycle of temporary tables.	1
biz_id	The user ID.	None
verbose	Specifies whether to display logs.	False
verbose_log	The log receiver.	None
chunk_size	The size of the write buffer.	1496
retry_times	The number of request retries.	4
pool_connections	The number of cached connections in the connection pool.	10
pool_maxsize	The maximum capacity of the connection pool.	10
connect_timeout	The connection timeout period (unit: seconds).	5
read_timeout	The read timeout period (unit: seconds).	120
api_proxy	The API proxy server.	None
data_proxy	The data proxy server.	None
completion_size	The limit on the number of object completion listing items.	10
notebook_repr_widget	Specifies whether to use interactive graphs.	True

Option	Description	Default value
sql.settings	MaxCompute SQL runs global hints.	None
sql.use_odps2_extension	Specifies whether to enable MaxCompute 2.0 language extension.	False

Data upload and download configuration

Option	Description	Default value
tunnel.endpoint	The endpoint of Tunnel.	None
tunnel.use_instance_tunnel	Specifies whether to use Instance Tunnel to obtain execution results.	True
tunnel.limit_instance_tunnel	Specifies whether to limit the number of data records obtained by using Instance Tunnel.	None
tunnel.string_as_binary	Specifies whether to use bytes instead of unicode for data of the STRING type.	False

DataFrame configuration

Option	Description	Default value
interactive	Specifies whether DataFrame is used in an interactive environment.	Depends on the detection value.
df.analyze	Specifies whether to enable non-MaxCompute built-in functions.	True
df.optimize	Specifies whether to enable full DataFrame optimization.	True
df.optimizes.pp	Specifies whether to enable DataFrame predicate push optimization.	True
df.optimizes.cp	Specifies whether to enable DataFrame column pruning optimization.	True
df.optimizes.tunnel	Specifies whether to enable DataFrame tunnel optimization.	True

Option	Description	Default value
df.quote	Specifies whether to use `` to mark fields and table names in the backend of MaxCompute SQL.	True
df.libraries	The third-party library (resource name) that is used by DataFrame.	None
df.supersede_libraries	Specifies whether to use the self-uploaded NumPy to replace the version in the service.	False
df.odps.sort.limit	The default limit on the number of items added during a sort operation of DataFrame.	10000

Machine learning configuration

Option	Description	Default value
ml.xflow_settings	The XFlow execution configuration.	None
ml.xflow_project	The default XFlow project name.	algo_public
ml.use_model_transfer	Specifies whether to use ModelTransfer to obtain the model PMML.	False
ml.model_volume	The name of the volume used by ModelTransfer.	pyodps_volume

1.11.9. API overview

Links to PyODPS API documentation are provided below. You can see the parameter description and example of each function:

- Definitions
- DataFrame Reference

1.11.10. FAQ

How do I view the current PyODPS version?

Run the following commands:

```
import odps
print(odps.__version__)
```

How do I troubleshoot the "Project not found" error?

This problem is generally caused by incorrect endpoint configurations. You must check the configurations and rectify the problem. You also need to check whether the position of the MaxCompute object parameter is correct.

How do I manually specify a Tunnel endpoint?

You can create your MaxCompute object with the tunnel_endpoint parameter specified, as shown in the following code. Replace the content enclosed with asterisks (*) with actual parameter values and remove the asterisks (*).

```
from odps import ODPS
o = ODPS('**your-access-id**', '**your-secret-access-key**', '**your-default-project**',
        endpoint='**your-end-point**', tunnel_endpoint='**your-tunnel-endpoint**')
```

How do I troubleshoot the "project is protected" error reported while data is being read?

The project security policy does not allow you to read data from tables. To retrieve all the data, you can use the following solutions:

- Contact the project owner to add exception rules.
- Use DataWorks or other masking tools to mask the data and then export the data to an unprotected project before reading it.

To retrieve part of the data, you can use the following solutions:

- Use `o.execute_sql('select * from <table_name>').open_reader()` .
- Use `o.get_table('<table_name>').to_df()` in DataFrame.

I can only retrieve a maximum of 10,000 data records by executing SQL command open_reader. How do I retrieve more than 10,000 data records?

Use `create table as select ...` to save the SQL result to a table, and then use `table.open_reader` to read data.

How do I troubleshoot the "ODPSError: ODPS entrance should be provided" error reported when I upload pandas DataFrame to MaxCompute?

This error is reported because the global MaxCompute object cannot be found. Use one of the following methods to resolve this problem:

- If you use the room mechanism `%enter` , configure the global MaxCompute object.

- Call the `to_global` method for the MaxCompute object.
- Use the following MaxCompute parameter:

```
DataFrame(pd_df).persist('your_table', odps=odps)
```

How do I use max_pt in DataFrame?

Use the `odps.df.func` module to call built-in functions of MaxCompute. Example:

```
from odps.df import func
df = o.get_table('your_table').to_df()
df[df.ds == func.max_pt('your_project.your_table')]  # ds is a partition column.
```

How do I troubleshoot the "table lifecycle is not specified in mandatory mode" error reported when I use DataFrame to write data to a table?

Your project requires that every table be created with a lifecycle. Therefore, you must make the following configuration every time you run your own code:

```
from odps import options
options.lifecycle = 7  # You can also set this parameter to your expected lifecycle in days.
```

How do I traverse each row of data in PyODPS DataFrame?

PyODPS DataFrame currently does not support this feature. PyODPS DataFrame focuses on handling large volumes of data. Traversing data is inefficient.

We recommend that you use the apply or map_reduce method of DataFrame to parallelize your serial traverse operations.

If you confirm that data traversing is necessary in your scenario and that the cost is acceptable, you can use the `to_pandas` method to convert your DataFrame to pandas DataFrame, or you can store DataFrame as a table and use `read_table` or Tunnel to read data.

1.12. Java sandbox limits

MaxCompute MapReduce and UDF programs in distributed environments are run in Java sandboxes. The main programs of MapReduce are not subject to these limits.

The limits include:

- Local files cannot be accessed directly, and can only be accessed through APIs provided by MaxCompute MapReduce or Graph. You can access the resources specified by the resources option, such as files, JAR packages, and resource tables. You can use System.out and System.err to export logs and run the log command on the MaxCompute console to view log information.
- Direct access to the distributed file system is not allowed. You can only use MaxCompute MapReduce to Graph to access records of tables.
- JNI calls are not allowed.

- Java threads cannot be created, and Linux commands cannot be executed by sub-threads.
- Network access, including the operation of getting the local IP address, is prohibited.
- Java reflection restriction: The suppressAccessChecks permission is prohibited, so you cannot set a private attribute or method accessible to read private attributes or call private methods.

  Specifically, an access denied error is returned when you perform any of the preceding operations.

  Methods for accessing local files:

  java.io.File:

```
public boolean delete()
public void deleteOnExit()
public boolean exists()
public boolean canRead()
public boolean isFile()
public boolean isDirectory()
public boolean isHidden()
public long lastModified()
public long length()
public String[] list()
public String[] list(FilenameFilter filter)
public File[] listFiles()
public File[] listFiles(FilenameFilter filter)
public File[] listFiles(FileFilter filter)
public boolean canWrite()
public boolean createNewFile()
public static File createTempFile(String prefix, String suffix)
public static File createTempFile(String prefix, String suffix,File directory)
public boolean mkdir()
public boolean mkdirs()
public boolean renameTo(File dest)
public boolean setLastModified(long time)
public boolean setReadOnly()
java.io.RandomAccessFile:
RandomAccessFile(String name, String mode)
RandomAccessFile(File file, String mode)
java.io.FileInputStream:
FileInputStream(FileDescriptor fdObj)
FileInputStream(String name) FileInputStream(File file)
java.io.FileOutputStream:
FileOutputStream(FileDescriptor fdObj)
FileOutputStream(File file)
```

```
FileOutputStream(String name)
FileOutputStream(String name, boolean append)
java.lang.Class:
public ProtectionDomain getProtectionDomain()
java.lang.ClassLoader:
ClassLoader ()
ClassLoader(ClassLoader parent)
java. lang. Runtime:
public Process exec(String command)
public Process exec(String command, String envp[])
public Process exec(String cmdarray[])
public Process exec(String cmdarray[], String envp[])
public void exit(int status)
public static void runFinalizersOnExit(boolean value)
public void addShutdownHook(Thread hook)
public boolean removeShutdownHook(Thread hook)
public void load(String lib)
public void loadLibrary(String lib)
java.lang.System:
public static void exit(int status)
public static void runFinalizersOnExit(boolean value)
public static void load(String filename)
public static void loadLibrary( String libname)
public static Properties getProperties()
public static void setProperties(Properties props)
public static String getProperty(String key)
// Only some keys are accessible.
public static String getProperty(String key, String def)
// Only some keys are accessible.
public static String setProperty(String key, String value)
public static void setIn(InputStream in)
public static void setOut(PrintStream out)
public static void setErr(PrintStream err)
public static synchronized void setSecurityManager(SecurityManager s)
```

List of keys allowed by System.getProperty:

```
java.version
java.vendor
java.vendor.url
java.class.version
os.name os.version
```

os.arch
file.separator
path.separator
line.separator
java.specification.version
java.specification.vendor
java.specification.name
java.vm.specification.version
java.vm.specification.vendor
java.vm.specification.name
java.vm.version
java.vm.vendor
java.vm.name
file.encoding
user.timezone
java. lang. Thread:
Thread ()
Thread(Runnable target)
Thread(String name)
Thread(Runnable target, String name)
Thread(ThreadGroup group, ...)
public final void checkAccess()
public void interrupt()
public final void suspend()
public final void resume()
public final void setPriority (int newPriority)
public final void setName(String name)
public final void setDaemon(boolean on)
public final void stop()
public final synchronized void stop(Throwable obj)
public static int enumerate(Thread tarray[])
public void setContextClassLoader(ClassLoader cl)
java. lang. ThreadGroup:
ThreadGroup (String name)
ThreadGroup(ThreadGroup parent, String name)
public final void checkAccess()
public int enumerate(Thread list[])
public int enumerate(Thread list[], boolean recurse)
public int enumerate(ThreadGroup list[])
public int enumerate(ThreadGroup list[], boolean recurse)
public final ThreadGroup getParent()

```
public final ThreadGroup getParent()
public final void setDaemon(boolean daemon)
public final void setMaxPriority(int pri)
public final void suspend()
public final void resume()
public final void destroy()
public final void interrupt()
public final void stop()
java. lang. reflect. AccessibleObject:
public static void setAccessible (...)
public void setAccessible (...)
java.net. InetAddress:
public String getHostName ()
public static InetAddress[] getAllByName(String host)
public static InetAddress getLocalHost()
java.net.DatagramSocket:
public InetAddress getLocalAddress()
java.net.Socket:
Socket(...)
java.net.ServerSocket:
ServerSocket (...)
public Socket accept()
protected final void implAccept(Socket s)
public static synchronized void setSocketFactory(...)
public static synchronized void setSocketImplFactory(...)
java.net.DatagramSocket:
DatagramSocket (...)
public synchronized void receive(DatagramPacket p)
java.net.MulticastSocket:
MulticastSocket(...)
java.net.URL:
URL(...)
public static synchronized void setURLStreamHandlerFactory(...)
java.net.URLConnection
public static synchronized void setContentHandlerFactory(...)
public static void setFileNameMap(FileNameMap map)
java.net.HttpURLConnection:
public static void setFollowRedirects(boolean set)
java.net.URLClassLoader
URLClassLoader(...)
java.security.AccessControlContext:
```

```
public AccessControlContext(AccessControlContext acc, DomainCombiner combiner)
public DomainCombiner getDomainCombiner()
```

1.13. Volume lifecycle management

1.13.1. Overview

In the previous version of MaxCompute, a partition in a volume does not have a lifecycle and can exist forever. You have to manage the lifecycle of a volume. In some cases, user management of volume lifecycle may cause a problem. For example, if the account used to delete a partition is different from the account used to create the partition, the delete operation fails. The new feature of volume lifecycle management can solve this problem.

For more information about simple volume lifecycle management operations, see *Volume lifecycle operations* .

1.13.2. Volume lifecycle operations

Create a volume with a specified lifecycle

Example:

```
odps@ your_project>fs -mkv test_volume -lifecycle 7 "this is a test volume";
OK
```

Modify the lifecycle of a volume

Example:

```
odps@ your_project>fs -alter test_volume -lifecycle 3;
OK
```

View the lifecycle of a volume

Example:

```
odps@ your_project>fs -meta test_volume;
Comment: "this is a test volume"
Length: 0
File number: 0
Lifecycle: 3
OK
```

1.14. Spark on MaxCompute

1.14.1. Overview

Spark on MaxCompute is a solution developed by Alibaba Cloud to enable the seamless use of Spark on the MaxCompute platform. It supplements a wide variety of features to MaxCompute.

Spark on MaxCompute provides a native Spark user experience and offers native Spark components and APIs. Spark on MaxCompute can access MaxCompute data sources and enhance security for multi-tenant scenarios. Spark on MaxCompute can also act as a management platform to share resources, storage, and user systems between jobs and ensure high performance at a low cost. Spark can work with MaxCompute to create better and more efficient data processing solutions. Community Spark applications can run in **Spark on MaxCompute**.

Spark on MaxCompute has an independent data development node in DataWorks and supports data development in DataWorks.

1.14.2. Project resources

Before you use **Spark on MaxCompute**, you may need to pay attention to or download the following project resources:

- **Spark on MaxCompute** release package: Download the latest release package at Spark on MaxCompute.
- **Spark on MaxCompute** plugin: This is an open-source program. Download the plugin at Aliyun Cupid SDK.

After you prepare the preceding project resources, you need to complete environment configuration. Then, you can run related GitHub demos.

1.14.3. Environment settings

1.14.3.1. Decompress the Spark on MaxCompute release package

Download the latest version of the **Spark on MaxCompute** release package and decompress it. The structure of the decompressed folder is as follows:

```
.
|-- R
|-- RELEASE
|-- __spark_libs__.zip
|-- bin
|-- conf
|-- cupid
|-- derby.log
|-- examples
|-- jars
|-- logs
|-- metastore_db
|-- python
|-- sbin
|-- yarn
```

1.14.3.2. Set environment variables

Set required environment variables.

> Note The main environment variables are JAVA_HOME and SPARK_HOME.

Set JAVA_HOME

```
export JAVA_HOME=/path/to/jdk
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH
```

Set SPARK_HOME

```
export SPARK_HOME=/path/to/spark_extracted_package
export PATH=$SPARK_HOME/bin:$PATH
```

If you use SparkR, install R in the */home/admin/R* directory. Then, run the following command to set the path:

```
export PATH=/home/admin/R/bin/:$PATH
```

If you use PySpark, install Python 2.7. Then, run the following command to set the path:

```
export PATH=/path/to/python/bin/:$PATH
```

1.14.3.3. Configure Spark-defaults.conf

The *$SPARK_HOME/conf* directory contains a file named spark-defaults.conf. Before you submit a Spark task to MaxCompute, you must configure your MaxCompute account in this file.

The following content is the default configuration in the file. You only need to enter your account information in the blanks.

```
# OdpsAccount Info Setting
spark.hadoop.odps.project.name=
spark.hadoop.odps.access.id=
spark.hadoop.odps.access.key=
spark.hadoop.odps.end.point=
#spark.hadoop.odps.moye.trackurl.host=
#spark.hadoop.odps.cupid.webproxy.endpoint=
spark.sql.catalogImplementation=odps

# spark-shell Setting
spark.driver.extraJavaOptions -Dscala.repl.reader=com.aliyun.odps.spark_repl.OdpsInteractiveReader
-Dscala.usejavacp=true

# SparkR Setting
# odps.cupid.spark.r.archive=/path/to/R-PreCompile-Package.zip

# Cupid Longtime Job
# spark.hadoop.odps.cupid.engine.running.type=longtime
# spark.hadoop.odps.cupid.job.capability.duration.hours=8640
# spark.hadoop.odps.moye.trackurl.dutation=8640

# spark.r.command=/home/admin/R/bin/Rscript
# spark.hadoop.odps.cupid.disk.driver.enable=false
spark.hadoop.odps.cupid.bearer.token.enable=false
spark.hadoop.odps.exec.dynamic.partition.mode=nonstrict
```

1.14.4. Quick start

This topic describes how to use **Spark on MaxCompute**.

1. Download the Spark package from Spark on MaxCompute and decompress the package.
2. Set environment variables.

```
export SPARK_HOME=/path/to/spark-2.1.0-private-cloud-v3.1.0
export JAVA_HOME=/path/to/java/
```

3. Set spark-defaults.conf.

```
cp $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-defaults.conf
```

Edit $SPARK_HOME/conf/spark-defaults.conf by filling information in the blanks.

```
# OdpsAccount Info Setting
spark.hadoop.odps.project.name=
spark.hadoop.odps.access.id=
spark.hadoop.odps.access.key=
spark.hadoop.odps.end.point=
#spark.hadoop.odps.moye.trackurl.host=
#spark.hadoop.odps.cupid.webproxy.endpoint=
spark.sql.catalogImplementation=odps

# spark-shell Setting
spark.driver.extraJavaOptions -Dscala.repl.reader=com.aliyun.odps.spark_repl.OdpsInteractiveRe
ader -Dscala.usejavacp=true

# SparkR Setting
# odps.cupid.spark.r.archive=/path/to/R-PreCompile-Package.zip

# Cupid Longtime Job
# spark.hadoop.odps.cupid.engine.running.type=longtime
# spark.hadoop.odps.cupid.job.capability.duration.hours=8640
# spark.hadoop.odps.moye.trackurl.dutation=8640

# spark.r.command=/home/admin/R/bin/Rscript
# spark.hadoop.odps.cupid.disk.driver.enable=false
spark.hadoop.odps.cupid.bearer.token.enable=false
spark.hadoop.odps.exec.dynamic.partition.mode=nonstrict
```

4. Prepare spark-example.

```
git clone https://github.com/aliyun/aliyun-cupid-sdk.git
cd aliyun-cupid-sdk
mvn -T 1C clean install -DskipTests
```

5. Run SparkPi.

```
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class com.aliyun.odps.spark.examples.SparkPi /path/to/
aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-shade
d.jar
```

If you see the following output, your operation is successful. Other logs may be included in the output.

```
18/02/09 15:52:28 INFO Client: Application report for application_1518162700322_1635034099 (state: FIN
ISHED)
18/02/09 15:52:28 INFO Client:
 client token: N/A
 diagnostics: N/A
 ApplicationMaster host: ***
 ApplicationMaster RPC port: ***
 queue: queue
 start time: 1518162732343
 final status: SUCCEEDED
 tracking URL: http://***:80/proxyview/jobview/?h=http://***:80/api&p=odps_smoke_test&i=20180209
075148695gbkp8e01&t=spark&id=application_1518162700322_1635034099&metaname=201802090751486
95gbkp8e01&token=YkhjNXJWZ0dvdzVScXVFQWpCQWMra1RSZHVFPSxPRFBTX09CTzoxMzY1OTM3MTUw
NzcyMjEzLDE1MTg0MjE5MzUseyJTdGF0ZW1lbnQiOlt7IkFjdGlvbiI6WyJvZHBzOlJlYWQiXSwiRWZmZWN0IjoiQ
Wxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnByb2plY3RzL29kcHNfc21va2VfdGVzdC9pbnN0YW5jZXM
vMjAxODAyMDkwNzUxNDg2OTVnYmtwOGUwMSJdfV0sIlZlcnNpb24iOiIxIn0=
 user: user
18/02/09 15:52:28 INFO ShutdownHookManager: Shutdown hook called
18/02/09 15:52:28 INFO ShutdownHookManager: Deleting directory /tmp/spark-d77416ad-79a8-49f7-93
1d-0533663b5d85
```

1.14.5. Demo

This topic provides a demo on how to use Spark.

The demo is as follows:

1. Modify the configuration file.

   Decompress the Spark package, go to the *conf* directory, and modify the following configuration items in the configuration file spark-defaults.conf:

   ```
   spark.hadoop.odps.project.name=xxxx
   spark.hadoop.odps.access.id=xxxx
   spark.hadoop.odps.access.key=xxxx
   spark.hadoop.odps.end.point=http://service.xxxx.xxxx.xxxxx.qd-inc.com:80/api
   spark.hadoop.odps.cupid.distributedcache.mincopy=3
   spark.hadoop.odps.cupid.distributedcache.maxcopy=3
   spark.hadoop.odps.cupid.proxy.domain.name=jobview.xxxx.xxxx.xxxx.com
   spark.hadoop.odps.cupid.history.server.address=10.xx.xx.xx:18080
   ```

> **Note**
>
> - You can obtain the first four configuration items from the web_component.conf file in the */cloud/app/odps-service-console/CupidFrontendServer#/cupid_web_proxy/current/conf.local/* directory.
> - spark.hadoop.odps.cupid.proxy.domain.name: specifies the domain name of odps_jobview_server_dns. Remove sparkui at the beginning of the domain name.
> - spark.hadoop.odps.cupid.history.server.address: specifies the IP address of the AG machine. Add the port number 18080 to the end of the IP address.

2. Start the program.

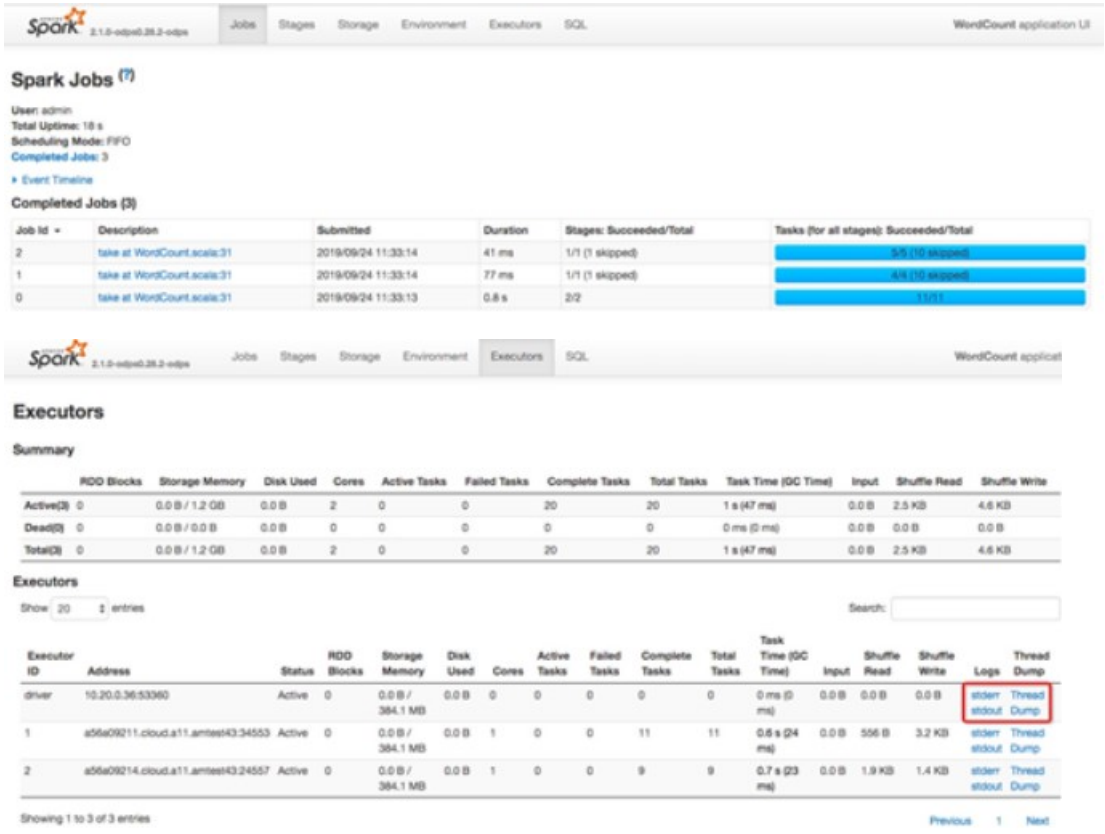
   Go to the directory where the Spark package is decompressed and run the following commands:

```
bin/spark-submit
--class com.aliyun.odps.spark.examples.WordCount --master yarn-cluster
examples/spark-examples-2.0.0-SNAPSHOT-shaded.jar
```

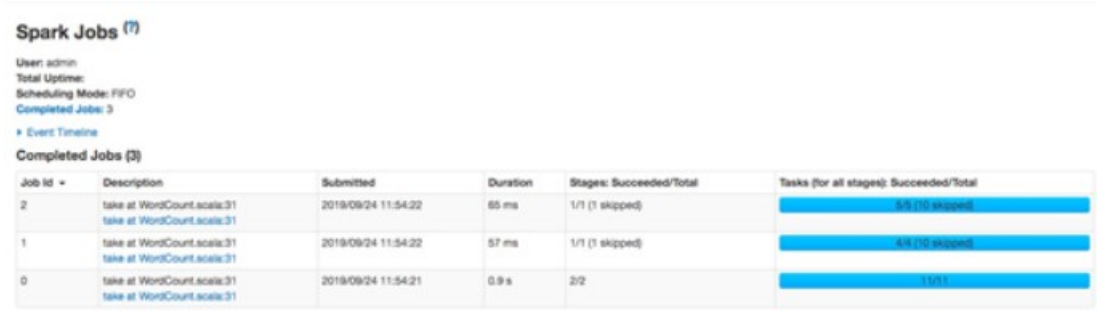
3. Access the link in the command output.

   LogView

SparkUI



Spark History



1.14.6. Common cases

1.14.6.1. WordCount example

Spark runs simple WordCount.

> **Note**
>
> You must download the GitHub project and compile the project before running the corresponding demos.
>
> ```
> git clone https://github.com/aliyun/aliyun-cupid-sdk.git
> -- Download a GitHub project.
> cd aliyun-cupid-sdk
> mvn -T 1C clean install -DskipTests
> -- Compile the GitHub project.
> ```
>
> After you complete the preceding steps, JAR packages are created. These JAR packages will be used to run the demos in this and the subsequent topics.

The following is the code for this example:

```
package com.aliyun.odps.spark.examples

import org.apache.spark.sql.SparkSession

object WordCount {
  def main(args: Array[String]) {
    val spark = SparkSession
      .builder()
      .appName("WordCount")
      .getOrCreate()
    val sc = spark.sparkContext
    try {
      sc.parallelize(1 to 100, 10).map(word => (word, 1)).reduceByKey(_ + _, 10).take(100).foreach(println
)
    } finally {
      sc.stop()
    }
  }
}
```

Run the following command to submit the job:

```
bin/spark-submit \
--master yarn-cluster \
--class com.aliyun.odps.spark.examples.WordCount \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-
SNAPSHOT-shaded.jar
```

1.14.6.2. OSS access example

Spark can access OSS data.

Example:

```
package com.aliyun.odps.spark.examples.oss

import org.apache.spark.sql.SparkSession

object SparkUnstructuredDataCompute {
 def main(args: Array[String]) {
   val spark = SparkSession
     .builder()
     .appName("SparkUnstructuredDataCompute")
     .config("spark.hadoop.fs.oss.accessKeyId", "***")
     .config("spark.hadoop.fs.oss.accessKeySecret", "***")
     .config("spark.hadoop.fs.oss.endpoint", "oss-cn-hangzhou-zmf.aliyuncs.com")
     .getOrCreate()

   val sc = spark.sparkContext
   try {
     val pathIn = "oss://bucket/inputdata/"
     val inputData = sc.textFile(pathIn, 5)
     val cnt = inputData.count
     println(s"count: $cnt")
   } finally {
     sc.stop()
   }
 }
}
```

Run the following command to submit the job:

```
./bin/spark-submit
--jars cupid/hadoop-aliyun-package-3.0.0-alpha2-odps-jar-with-dependencies.jar
--class com.aliyun.odps.spark.examples.oss.SparkUnstructuredDataCompute
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar
```

1.14.6.3. MaxCompute table read/write example

Read/write a MaxCompute table and convert it to Spark RDD.

> **Notice** The project or table specified in the demo must exist or be changed to the specific project or table.

Example:

```
package com.aliyun.odps.spark.examples

import com.aliyun.odps.data.Record
import com.aliyun.odps.{ PartitionSpec, TableSchema}
import org.apache.spark.odps.OdpsOps
import org.apache.spark.sql.SparkSession
import scala.util.Random

object OdpsTableReadWrite {
 def main(args: Array[String]) {

   val spark = SparkSession
    .builder()
    .appName("OdpsTableReadWrite")
    .getOrCreate()

   val sc = spark.sparkContext
   val projectName = sc.getConf.get("odps.project.name")

   try {
    val odpsOps = new OdpsOps(sc)

    // read from normal table via rdd api
    val rdd_0 = odpsOps.readTable(
     projectName,
     "cupid_wordcount",
     (r: Record, schema: TableSchema) => (r.getString(0), r.getString(1))
```

```
    )

    //read from single partition column table via rdd api
    val rdd_1 = odpsOps.readTable(
      projectName,
      "dftest_single_parted",
      Array("pt=20160101"),
      (r: Record, schema: TableSchema) => (r.getString(0), r.getString(1), r.getString("pt"))
    )

    // read from multi partition column table via rdd api
    val rdd_2 = odpsOps.readTable(
      projectName,
      "dftest_parted",
      Array("pt=20160101,hour=12"),
      (r: Record, schema: TableSchema) => (r.getString(0), r.getString(1), r.getString("pt"), r.getString(3
))
    )

    // read with multi partitionSpec definition via rdd api
    val rdd_3 = odpsOps.readTable(
      projectName,
      "cupid_partition_table1",
      Array("pt1=part1,pt2=part1", "pt1=part1,pt2=part2", "pt1=part2,pt2=part3"),
      (r: Record, schema: TableSchema) => (r.getString(0), r.getString(1), r.getString("pt1"), r.getString(
"pt2"))
    )

    // save rdd into normal table
    val transfer_0 = (v: Tuple2[String, String], record: Record, schema: TableSchema) => {
      record.set("id", v._1)
      record.set(1, v._2)
    }
    odpsOps.saveToTable(projectName, "cupid_wordcount_empty", rdd_0, transfer_0, true)

    // save rdd into partition table with single partition spec
    val transfer_1 = (v: Tuple2[String, String], record: Record, schema: TableSchema) => {
      record.set("id", v._1)
      record.set("value", v._2)
    }
    odpsOps.saveToTable(projectName, "cupid_partition_table1", "pt1=test,pt2=dev", rdd_0, transfer_1
```

```
    odpsOps.saveToTable(projectName, "cupid_partition_table1", "pt1=test,pt2=dev", rdd_0, transfer_1
, true)

    // dynamic save rdd into partition table with multiple partition spec
    val transfer_2 = (v: Tuple2[String, String], record: Record, part: PartitionSpec, schema: TableSchem
a) => {
      record.set("id", v._1)
      record.set("value", v._2)

      val pt1_value = if (new Random().nextInt(10) % 2 == 0) "even" else "odd"
      val pt2_value = if (new Random().nextInt(10) % 2 == 0) "even" else "odd"
      part.set("pt1", pt1_value)
      part.set("pt2", pt2_value)
    }
    odpsOps.saveToTableForMultiPartition(projectName, "cupid_partition_table1", rdd_0, transfer_2, tr
ue)
   } catch {
    case ex: Exception => {
      throw ex
    }
   } finally {
    sc.stop
   }
 }
}
```

Run the following command to submit the job:

```
bin/spark-submit \
--master yarn-cluster \
--class com.aliyun.odps.spark.examples.OdpsTableReadWrite \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-s
haded.jar
```

You can use either of the following methods to adjust the MaxCompute table read concurrency:

- Modify the spark.hadoop.odps.input.split.size parameter. A larger value results in a smaller number of Map tasks. The default value is 256 MB.
- Set numPartition in OdpsOps.readTable. The value determines the number of Map tasks. The number is calculated based on spark.hadoop.odps.input.split.size.

1.14.6.4. MaxCompute Table Spark-SQL example

Use sqlContext to read/write a MaxCompute table.

> **Notice**
>
> - The project or table specified in the demo must exist or be changed to the specific project or table.
> - Spark-defaults.conf must contain the setting *spark.sql.catalogImplementation = odps*.

Example:

```
package com.aliyun.odps.spark.examples

import org.apache.spark.sql.SparkSession

object OdpsTableReadWriteViaSQL {

 def main(args: Array[String]) {

  // please make sure spark.sql.catalogImplementation=odps in spark-defaults.conf
  // to enable odps catalog
  val spark = SparkSession
   .builder()
   .appName("OdpsTableReadWriteViaSQL")
   .getOrCreate()

  val projectName = spark.sparkContext.getConf.get("odps.project.name")
  val tableName = "cupid_wordcount"

  // get a ODPS table as a DataFrame
  val df = spark.table(tableName)
  println(s"df.count: ${df.count()}")

  // Just do some query
  spark.sql(s"select * from $tableName limit 10").show(10, 200)
  spark.sql(s"select id, count(id) from $tableName group by id").show(10, 200)

  // any table exists under project could be use
  // productRevenue
  spark.sql(
   """
    |SELECT product,
    |     category,
    |     revenue
```

```
      |FROM
      |  (SELECT product,
      |          category,
      |          revenue,
      |          dense_rank() OVER (PARTITION BY category
      |                             ORDER BY revenue DESC) AS rank
      |   FROM productRevenue) tmp
      |WHERE rank <= 2
    """.stripMargin).show(10, 200)

    spark.stop()
  }
}
```

Run the following command to submit the job:

```
bin/spark-submit \
--master yarn-cluster \
--class com.aliyun.odps.spark.examples.OdpsTableReadWrite \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-s
haded.jar
```

1.14.6.5. MaxCompute self-developed Console mode example

For safety reasons, machines in MaxCompute can not be directly connected. Therefore, the yarn-clientmode in native Spark cannot be used. To enable interaction, the MaxCompute team developed a proprietary client mode.

Example:

```
package com.aliyun.odps.spark.examples

import com.aliyun.odps.cupid.client.spark.client.CupidSparkClientRunner

object SparkClientNormalFT {
  def main(args: Array[String]) {
    val cupidSparkClient = CupidSparkClientRunner.getReadyCupidSparkClient()
    val jarPath = args(0) //client-jobexamples jar path
    val sparkClientNormalApp = new SparkClientNormalApp(cupidSparkClient)
    sparkClientNormalApp.runNormalJob(jarPath)
    cupidSparkClient.stopRemoteDriver()
  }
}
```

Run the following command to submit the job:

```
java -cp \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar:$SPARK_HOME/jars/* \
com.aliyun.odps.spark.examples.SparkClientNormalFT /path/to/aliyun-cupid-sdk/examples/client-jobexamples/target/client-jobexamples_2.11-1.0.0-SNAPSHOT.jar
```

1.14.6.6. MaxCompute Table PySpark example

Use PySpark to read/write MaxCompute tables.

Example:

```
from odps.odps_sdk import OdpsOps
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext, DataFrame

if __name__ == '__main__':
    conf = SparkConf().setAppName("odps_pyspark")
    sc = SparkContext(conf=conf)
    sql_context = SQLContext(sc)

    project_name = "cupid_testa1"
    in_table_name = "cupid_wordcount"
    out_table_name = "cupid_wordcount_py"

    normal_df = OdpsOps.read_odps_table(sql_context, project_name, in_table_name)

    for i in normal_df.sample(False, 0.01).collect():
        print i
    print "Read normal odps table finished"

    OdpsOps.write_odps_table(sql_context, normal_df.sample(False, 0.001), project_name, out_table_na
me)
    print "Write normal odps table finished"
```

Run the following command to submit the job:

```
spark-submit \
--master yarn-cluster \
--jars /path/to/aliyun-cupid-sdk/external/cupid-datasource/target/cupid-datasource_2.11-1.0.0-SNAP
SHOT.jar \
--py-files /path/to/aliyun-cupid-sdk/examples/spark-examples/src/main/python/odps.zip \
/path/to/aliyun-cupid-sdk/examples/spark-examples/src/main/python/odps_table_rw.py
```

1.14.6.7. Mllib example

We recommend that you use OSS for read/write operations in the Mllib model.

Example:

```
package com.aliyun.odps.spark.examples.mllib

import org.apache.spark.mllib.clustering.KMeans._
import org.apache.spark.mllib.clustering.{ KMeans, KMeansModel}
```

```
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.sql.SparkSession

object KmeansModelSaveToOss {
 val modelOssDir = "oss://bucket/kmeans-model"

 def main(args: Array[String]) {

  //1. train and save the model
  val spark = SparkSession
   .builder()
   .appName("KmeansModelSaveToOss")
   .config("spark.hadoop.fs.oss.accessKeyId", "***")
   .config("spark.hadoop.fs.oss.accessKeySecret", "***")
   .config("spark.hadoop.fs.oss.endpoint", "***")
   .getOrCreate()

  val sc = spark.sparkContext
  val points = Seq(
   Vectors.dense(0.0, 0.0),
   Vectors.dense(0.0, 0.1),
   Vectors.dense(0.1, 0.0),
   Vectors.dense(9.0, 0.0),
   Vectors.dense(9.0, 0.2),
   Vectors.dense(9.2, 0.0)
  )
  val rdd = sc.parallelize(points, 3)
  val initMode = K_MEANS_PARALLEL
  val model = KMeans.train(rdd, k = 2, maxIterations = 2, runs = 1, initMode)
  val predictResult1 = rdd.map(feature => "cluster id: " + model.predict(feature) + " feature:" + featur
e.toArray.mkString(",")).collect
  println("modelOssDir=" + modelOssDir)
  model.save(sc, modelOssDir)

  //2. predict from the oss model
  val modelLoadOss = KMeansModel.load(sc, modelOssDir)
  val predictResult2 = rdd.map(feature => "cluster id: " + modelLoadOss.predict(feature) + " feature:"
+ feature.toArray.mkString(",")).collect
  assert(predictResult1.size == predictResult2.size)
  predictResult2.foreach(result2 => assert(predictResult1.contains(result2)))
 }
```

```
}
```

Run the following command to submit the job:

```
./bin/spark-submit
--jars cupid/hadoop-aliyun-package-3.0.0-alpha2-odps-jar-with-dependencies.jar
--class com.aliyun.odps.spark.examples.mllib.KmeansModelSaveToOss
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar
```

1.14.6.8. PySpark interactive execution example

PySpark can only run on a host that can be directly connected to a computing cluster.

Install Python 2.7 and set the following path:

```
export PATH=/path/to/python/bin/:$PATH
```

Start command:

```
bin/pyspark --master yarn-client
```

Interactive execution:

```
df=spark.sql("select * from spark_user_data")
df.show()
```

1.14.6.9. Spark-shell interactive execution example (read tables)

Spark-shell can only run on a host that can be directly connected to a computing cluster.

Start command:

```
bin/spark-shell --master yarn
```

Interactive execution:

```
sc.parallelize(0 to 100, 2).collect
sql("show tables").show
sql("select * from spark_user_data").show(200,100)
```

1.14.6.10. Spark-shell interactive execution example (MLlib and OSS read/write)

Spark-shell can only run on a host that can be directly connected to a computing cluster.

Add the following configuration to conf/spark-defaults.conf:

```
spark.hadoop.fs.oss.accessKeyId=***
spark.hadoop.fs.oss.accessKeySecret=***
spark.hadoop.fs.oss.endpoint=***
```

Start command:

```
bin/spark-shell --master yarn --jars cupid/hadoop-aliyun-package-3.0.0-alpha2-odps-jar-with-dependencies.jar
```

Interactive execution:

```
import org.apache.spark.mllib.clustering.KMeans._
import org.apache.spark.mllib.clustering.{ KMeans, KMeansModel}
import org.apache.spark.mllib.linalg.Vectors
val modelOssDir = "oss://your_bucket/kmeans-model"
val points = Seq(
    Vectors.dense(0.0, 0.0),
    Vectors.dense(0.0, 0.1),
    Vectors.dense(0.1, 0.0),
    Vectors.dense(9.0, 0.0),
    Vectors.dense(9.0, 0.2),
    Vectors.dense(9.2, 0.0)
  )
val rdd = sc.parallelize(points, 3)
val initMode = K_MEANS_PARALLEL
val model = KMeans.train(rdd, k = 2, maxIterations = 2, runs = 1, initMode)
val predictResult1 = rdd.map(feature => "cluster id: " + model.predict(feature) + " feature:" + feature.toArray.mkString(",")).collect
println("modelOssDir=" + modelOssDir)
model.save(sc, modelOssDir)
val modelLoadOss = KMeansModel.load(sc, modelOssDir)
val predictResult2 = rdd.map(feature => "cluster id: " + modelLoadOss.predict(feature) + " feature:" + feature.toArray.mkString(",")).collect
assert(predictResult1.size == predictResult2.size)
predictResult2.foreach(result2 => assert(predictResult1.contains(result2)))
```

1.14.6.11. SparkR interactive execution example

SparkR can only be run on a host which can directly connect to a computing cluster. In addition, you must install R in the */home/admin/R* directory and set the path:

```
export PATH=/home/admin/R/bin/:$PATH
```

Start command:

```
bin/sparkR --master yarn -—archives . /R/R.zip
```

Interactive execution:

```
df <- as.DataFrame(faithful)
df
head(select(df, df$eruptions))
head(select(df, "eruptions"))
head(filter(df, df$waiting < 50))

results <- sql("FROM spark_user_data SELECT *")
head(results)
```

1.14.6.12. GraphX-PageRank example

Spark on MaxCompute supports native GraphX.

Example:

```
package com.aliyun.odps.spark.examples.graphx

import org.apache.spark.{ SparkConf, SparkContext}
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

object PageRank {
 def main(args: Array[String]): Unit = {
  val conf = new SparkConf().setAppName("pagerank")
  val sc = new SparkContext(conf)

  // construct vertices
  val users: RDD[(VertexId, Array[String])] = sc.parallelize(List(
   "1,BarackObama,Barack Obama",
   "2,ladygaga,Goddess of Love",
   "3,jeresig,John Resig",
```

```
    "4,justinbieber,Justin Bieber",
    "6,matei_zaharia,Matei Zaharia",
    "7,odersky,Martin Odersky",
    "8,anonsys"
  ).map(line => line.split(",")).map(parts => (parts.head.toLong, parts.tail))

  // construct edges
  val followers: RDD[Edge[Double]] = sc.parallelize(Array(
    Edge(2L,1L,1.0),
    Edge(4L,1L,1.0),
    Edge(1L,2L,1.0),
    Edge(6L,3L,1.0),
    Edge(7L,3L,1.0),
    Edge(7L,6L,1.0),
    Edge(6L,7L,1.0),
    Edge(3L,7L,1.0)
  ))

  // construct graph
  val followerGraph: Graph[Array[String], Double] = Graph(users, followers)

  // restrict the graph to users with usernames and names
  val subgraph = followerGraph.subgraph(vpred = (vid, attr) => attr.size == 2)

  // compute PageRank
  val pageRankGraph = subgraph.pageRank(0.001)

  // get attributes of the top pagerank users
  val userInfoWithPageRank = subgraph.outerJoinVertices(pageRankGraph.vertices) {
    case (uid, attrList, Some(pr)) => (pr, attrList.toList)
    case (uid, attrList, None) => (0.0, attrList.toList)
  }

  println(userInfoWithPageRank.vertices.top(5)(Ordering.by(_. _2. _1)).mkString("\n"))
 }
}
```

Run the following command to submit the job:

```
bin/spark-submit \
--master yarn-cluster \
--class com.aliyun.odps.spark.examples.graphx.PageRank \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-shaded.jar
```

1.14.6.13. Spark Streaming - NetworkWordCount example

Spark on MaxCompute supports native Spark Streaming. To use NetworkWordCount, you need to install Netcat on your local host, and then run the following command:

```
$ nc -lk 9999
```

The input on the console then becomes the input of Spark Streaming.

Example:

```
package com.aliyun.odps.spark.examples.streaming

import org.apache.spark.SparkConf
import org.apache.spark.examples.streaming.StreamingExamples
import org.apache.spark.storage.StorageLevel
import org.apache.spark.streaming.{ Seconds, StreamingContext}

object NetworkWordCount {
 def main(args: Array[String]) {
   if (args.length < 2) {
    System.err.println("Usage: NetworkWordCount <hostname> <port>")
    System.exit(1)
   }

   StreamingExamples.setStreamingLogLevels()

   // Create the context with a 1 second batch size
   val sparkConf = new SparkConf().setAppName("NetworkWordCount")
   val ssc = new StreamingContext(sparkConf, Seconds(1))

   // Create a socket stream on target ip:port and count the
   // words in input stream of \n delimited text (eg. generated by 'nc')
   // Note that no duplication in storage level only for running locally.
   // Replication necessary in distributed scenario for fault tolerance.
   val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
   val words = lines.flatMap(_.split(" "))
   val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
   wordCounts.print()
   ssc.start()
   ssc.awaitTermination()
 }
}
```

Run the following command to submit the job:

```
bin/spark-submit \
--master local[4] \
--class com.aliyun.odps.spark.examples.streaming.NetworkWordCount \
/path/to/aliyun-cupid-sdk/examples/spark-examples/target/spark-examples_2.11-1.0.0-SNAPSHOT-s
haded.jar localhost 9999
```

1.14.7. Maven dependencies

The GitHub project mentioned earlier can be used as your quick start template. For custom development, use the following pom.xml file.

Use Spark community edition 2.3.0 and ensure that the scope is provided.

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.11</artifactId>
  <version>2.3.0</version>
  <scope>provided</scope>
</dependency>
```

The MaxCompute plugin has been released to the Maven warehouse. Add the following dependencies:

```
<dependency>
   <groupId>com.aliyun.odps</groupId>
   <artifactId>cupid-core_2.11</artifactId>
   <version>1.0.0</version>
   <scope>provided</scope>
</dependency>

<dependency>
   <groupId>com.aliyun.odps</groupId>
   <artifactId>cupid-client_2.11</artifactId>
   <version>1.0.0</version>
</dependency>

<dependency>
   <groupId>com.aliyun.odps</groupId>
   <artifactId>cupid-datasource_2.11</artifactId>
   <version>1.0.0</version>
</dependency>
```

The file list in the Maven warehouse is as follows:

- The core code of the Cupid platform encapsulates the Cupid task submission API and the parent-child process read/write table API.

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>cupid-core_2.11</artifactId>
    <version>1.0.0</version>
</dependency>
```

- The datasource encapsulates the spark-related MaxCompute table read/write API.

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>cupid-datasource_2.11</artifactId>
    <version>1.0.0</version>
```

- The SDK encapsulates the Cupid client mode.

```
<dependency>
    <groupId>com.aliyun.odps</groupId>
    <artifactId>cupid-client_2.11</artifactId>
    <version>1.0.0</version>
</dependency>
```

1.14.8. Special notes

1.14.8.1. Running modes

Spark on MaxCompute supports three running modes: Local, Cluster, and DataWorks.

Local mode

The Local mode is used to facilitate code debugging for applications. In Local mode, you can use **Spark on MaxCompute** the same way as native Spark in the community. Additionally, you can use Tunnel to read and write data from and to MaxCompute tables.

In this mode, you can use either an IDE or the command line to run Spark on MaxCompute. If you use this mode, you must add the `spark.master=local[N]` configuration. N indicates the CPU resources required to implement this mode.

To use Tunnel to read and write data from and to tables in Local mode, you must add the Tunnel configuration item to Spark-defaults.conf. Enter the endpoint based on the region and network environment where the MaxCompute project is located.

The following code provides an example on how to use the command line to run Spark on MaxCompute in this mode:

```
1.bin/spark-submit --master local[4] \
--class com.aliyun.odps.spark.examples.SparkPi \
${path to aliyun-cupid-sdk}/spark/spark-2.x/spark-examples/target/spark-examples_2.11-version-shaded.jar
```

Cluster mode

In Cluster mode, you need to specify the Main method as the entry point of a custom application. The Spark job ends when Main succeeds or fails. This mode is suited for offline jobs. You can use Spark on MaxCompute in this mode together with DataWorks to schedule jobs.

The following code provides an example on how to use the command line to run Spark on MaxCompute in this mode:

```
1.bin/spark-submit --master yarn-cluster \
-class SparkPi \
${ProjectRoot}/spark/spark-2.x/spark-examples/target/spark-examples_2.11-version-shaded.jar
```

DataWorks mode

You can run offline jobs of **Spark on MaxCompute** (in Cluster mode) in DataWorks to integrate and schedule the other types of nodes.

The following procedure is for reference only.

1. Upload the resources in the DataWorks business flow and click **Submit**.
2. In the created business flow, select **ODPS Spark** from **Data Analytics**.
3. Double-click the Spark node and define the Spark job.

   Select a Spark version, a development language, and a resource file. The resource file is the JAR file uploaded and published in the business flow.

   You can specify configuration items, such as the number of executors and memory size, for the job to be submitted.

   You also need to set `spark.hadoop.odps.cupid.webproxy.endpoint` to the endpoint of the region where the project is located, for example, http://service.cn.maxcompute.aliyun-inc.com/api.
4. You can manually run the Spark node to view the task operation log and obtain the URLs of LogView and JobView from the log for further analysis and diagnosis.
5. After you have defined the Spark job, you can orchestrate and schedule services of different types in the business flow.

1.14.8.2. Streaming tasks

MaxCompute also supports Spark Streaming. To support long-running tasks, add the following special configurations to spark-defaults.conf.

Compared with offline jobs, streaming jobs have additional configurations, which take effect immediately after they are completed in spark-defaults.conf.

```
spark.hadoop.odps.cupid.engine.running.type=longtime
-- Set the type to longtime so that the job will not be reclaimed.
spark.hadoop.odps.cupid.job.capability.duration.hours=25920
-- Set the duration.
spark.yarn.maxAppAttempts=10
-- Set the maximum number of retries for a failover.
spark.streaming.receiver.writeAheadLog.enable=true
-- Determine whether to enable the writeAheadLog mode. This mode prevents data loss but lowers th
e efficiency.
```

1.14.8.3. Job diagnosis

After you submit a job, you need to check the job log to determine whether the job is submitted and executed properly. MaxCompute provides LogView and Spark Web UI for you to diagnose jobs.

Submit a job in Spark-submit mode. Logs are also generated when you use DataWorks to execute Spark tasks. Example:

```
cd $SPARK_HOME
bin/spark-submit --master yarn-cluster --class  SparkPi /tmp/spark-2.x-demo/target/AliSpark-2.x-quic
kstart-1.0-SNAPSHOT-shaded.jar
```

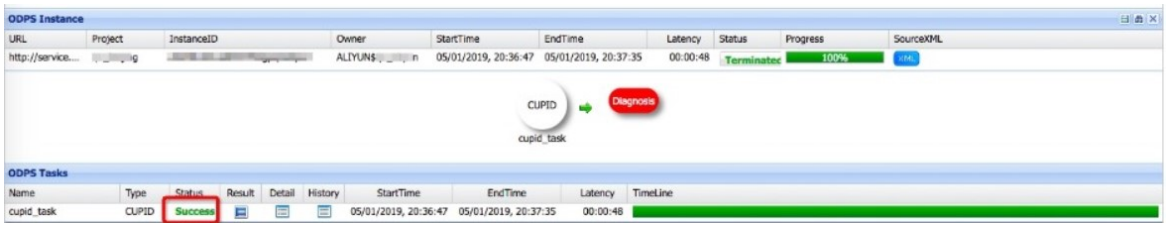
After the job is submitted, MaxCompute creates an instance and displays the LogView URL of the instance in the log.

```
19/01/05 20:36:47 INFO YarnClientImplUtil: logview url: http://logview.odps.aliyun.com/logview/?h=htt
p://service.cn.maxcompute.aliyun.com/api&p=qn_beijing&i=xxx&token=xxx
Success criterion: <If the following output is displayed, the operation succeeded. Other logs may be in
cluded in the output.>
19/01/05 20:37:34 INFO Client:
  client token: N/A
  diagnostics: N/A
  ApplicationMaster host: 11.220.xxx.xxx
  ApplicationMaster RPC port: 30002
  queue: queue
  start time: 1546691807945
  final status: SUCCEEDED
  tracking URL: http://jobview.odps.aliyun.com/proxyview/jobview/?h=http://service.cn.maxcompute.
aliyun-inc.com/api&p=project_name&i=xxx&t=spark&id=application_xxx&metaname=xxx&token=xxx
```

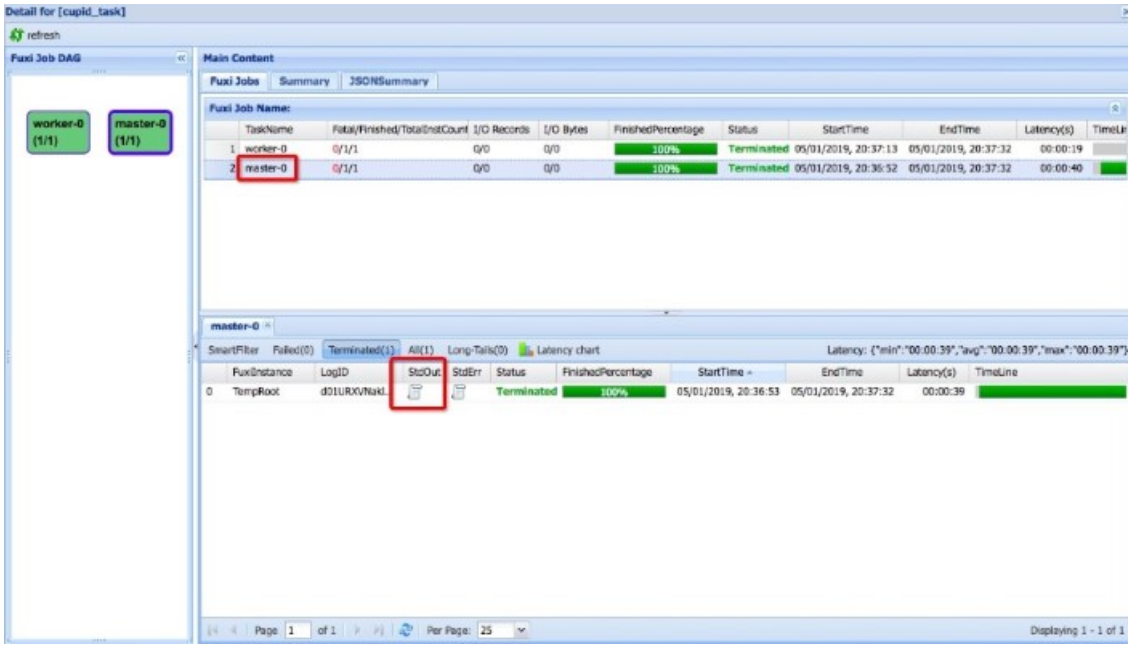
Use LogView to diagnose a job

Example:

1. View basic execution information about the task of the CUPID type in a browser based on the LogView URL.



2. Click the progress bar of the task whose **TaskName** is master-0. In the lower pane, click **All** and find TempRoot in the **FuxiInstance** column.



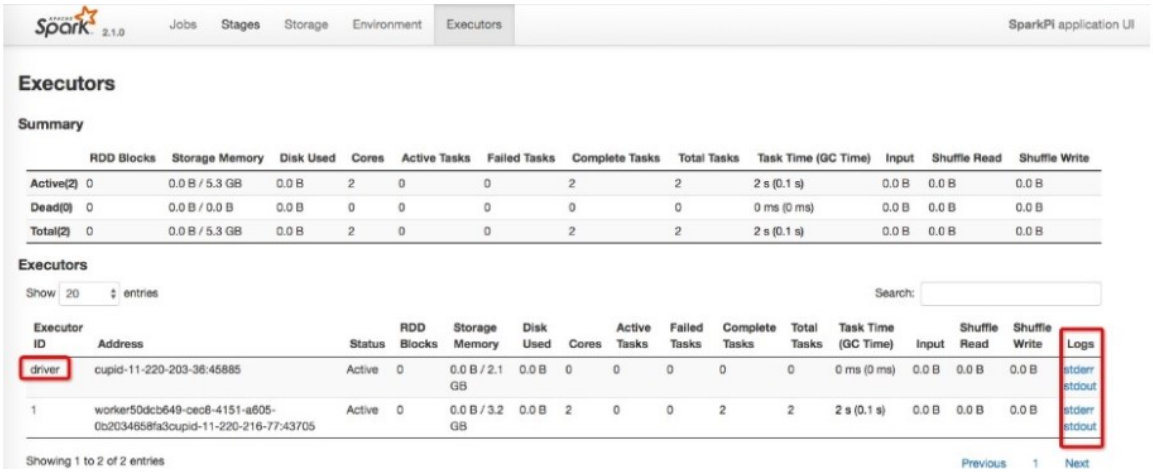
3. Click the icon in the **StdOut** column corresponding to TempRoot to view the output of SparkPi.

Use Spark Web UI to diagnose a job

The tracking URL in the log indicates that your job is submitted to the MaxCompute cluster. This URL is crucial because it is the URL of both Spark Web UI and History Server.

Example:

1. Access the URL in a browser to track the running status of your Spark job.



2. Click **stdout** in the Logs column corresponding to driver to view the output of the Spark job.

1.14.9. APIs supported by Spark

1.14.9.1. Spark Shell

Run the following commands to start the application.

```
$cd $SPARK_HOME
-- Access the spark directory.
<b>Start command</b>: <codeblock>bin/spark-shell --master yarn</codeblock>
-- Select a running mode and start the application.
```

Example:

```
sc.parallelize(0 to 100, 2).collect
sql("show tables").show
sql("select * from spark_user_data").show(200,100)
```

1.14.9.2. Spark R

Run the following commands to start the application:

```
$mkdir -p /home/admin/R && unzip . /R/R.zip -d /home/admin/R/
-- Create directory R and decompress R.zip in the directory.
$export PATH=/home/admin/R/bin/:$PATH
-- Set environment variables.
$bin/sparkR --master yarn --archives . /R/R.zip
-- Select a running mode and start the application.
```

Example:

```
df <- as.DataFrame(faithful)
df
head(select(df, df$eruptions))
head(select(df, "eruptions"))
head(filter(df, df$waiting < 50))
results <- sql("FROM spark_user_data SELECT *")
head(results)
```

1.14.9.3. Spark SQL

Run the following commands to start the application:

```
$cd $SPARK_HOME
-- Access the spark directory.
$bin/spark-sql --master yarn
-- Select a running mode and start the application.
```

Example:

```
show tables;
select * from spark_user_data limit 3;;
quit;
```

1.14.9.4. Spark JDBC

Run the following commands to start an application:

```
$sbin/stop-thriftserver.sh
-- Stop a thread.
$sbin/start-thriftserver.sh
-- Restart a thread.
$bin/beeline
-- Start an application.
```

Example:

```
! connect jdbc:hive2://localhost:10000/odps_smoke_test
show tables;
select * from mr_input limit 3;
! quit
```

1.14.10. Spark dynamic resource allocation

Background

Spark provides a large number of configuration items to implement a wide array of semantics. You can use the default values for most configuration items, but there are some items require manual configurations. Of these items, **spark.executor.instances** is the most complicated.

A value that is too small will cause operations to run slowly or fail, while a value that is too large will waste resources.

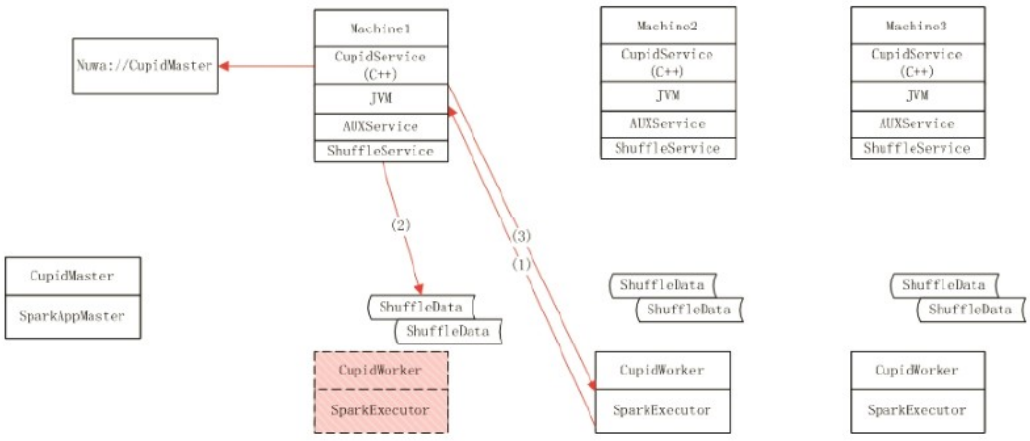
Even the optimal value based on full understanding of the data and logic is not reliable. For complex jobs, the number of executors required at different stages is different and different resources are required during the job execution process. Fixed configurations of resources will cause a waste. When long tail latency occurs, idle resources will be occupied by other executors even for simple jobs.

Solution

The best solution is to allocate resources as needed. Dynamic Resource Allocation (DRA) is such a solution. The CupidService developed by the Cupid team supports native DRA. Its implementation process is as shown in the following figure.

> **Note** For more information about the community native DRA and related configurations, see **Dynamic Resource Allocation** in **Job Scheduling** and **Dynamic Allocation** in **Spark Configuration**.

Implementation process



Enable DRA

You must add the following settings to enable DRA. You do not need to modify the code.

```
spark.hadoop.odps.cupid.shuffleservice.enable=true  // Required, the flag at the Cupid end.
spark.hadoop.odps.cupid.disk.driver.enable=true  // Required, the dependent item.
spark.dynamicAllocation.enabled=true // Required, the flag at the Spark end.
spark.shuffle.service.enabled=true // Required, the dependent item.
spark.shuffle.service.port=7338 // Required, the shuffle service port.
spark.authenticate=true // Required, authentication.
spark.dynamicAllocation.maxExecutors=128 // Optional, the maximum number of executors.
spark.dynamicAllocation.minExecutors=1 // Optional, the minimum number of executors.
spark.dynamicAllocation.initialExecutors=1 // Optional, the initial number of executors.
spark.dynamicAllocation.executorIdleTimeout=60s // Optional, the waiting period before idle executor
s are released.
```

When DRA is enabled, **spark.executor.instances** is optional and equivalent to `spark.dynamicAllocation.initialExecutors`.

1.14.11. FAQ

This topic describes the frequently asked questions about **Spark on MaxCompute**.

How do I migrate open-source Spark code to Spark on MaxCompute?

The migration method depends on whether the job needs to access MaxCompute tables or OSS:

- If the job does not need to access MaxCompute tables or OSS:

  Run the JAR package directly. Note that you must set the dependency on Spark or Hadoop to provided.

- If the job needs to access MaxCompute tables:

  Configure the related dependencies and re-package the application.

- If the job needs to access OSS:

  Configure the related dependencies and re-package the application.

How do I use Spark to access services in a VPC?

Currently, Spark does not allow access to services in a VPC. If you need to access a service in a VPC, submit a ticket to contact the MaxCompute team.

How do I troubleshoot the error indicating the ID and key provided by spark-defaults.conf are incorrect?

Error:

```
Stack:
com.aliyun.odps.OdpsException: ODPS-0410042:
Invalid signature value - User signature dose not match
```

Check whether the ID and key provided by spark-defaults.conf are consistent with the **AccessKey ID** and **AccessKey secret** obtained from the Apsara Stack console.

How do I troubleshoot the error indicating that I do not have permissions?

Error:

```
Stack:
com.aliyun.odps.OdpsException: ODPS-0420095:
Access Denied - Authorization Failed [4019], You have NO privilege 'odps:CreateResource' on {acs:odp
s:*:projects/*}
```

Contact the project owner to grant you the permissions to read and create resources.

How do I troubleshoot the error indicating that the project does not support Spark tasks?

Error:

```
Exception in thread "main" org.apache.hadoop.yarn.exceptions.YarnException: com.aliyun.odps.OdpsE
xception: ODPS-0420095: Access Denied - The task is not in release range: CUPID
```

Check whether the **Spark on MaxCompute** service is provided in the region where the project is located. Check whether the configuration of Spark-defaults.conf is consistent with the requirements in the service documentation. If the Spark on MaxCompute service is provided in the region and the configuration of Spark-defaults.conf is correct, submit a ticket.

How do I handle the "No space left on device" error reported while a task is running?

Error:

```
No space left on device
```

Spark uses online storage to replace local storage. Shuffled data and overflow data of BlockManager are all stored online. Therefore, you must check the setting of the online storage space. The online storage space is specified by the `spark.hadoop.odps.cupid.disk.driver.device_size` parameter. The default space is 20 GB and the maximum space is 100 GB. If this error persists after you adjust the capacity to 100 GB, further analysis is needed.

The most common cause is data skew. During the shuffle and cache processes, data is centrally distributed in some blocks. In this case, you can decrease the number of concurrent tasks in each executor (spark.executor.cores) or increase the number of executors (spark.executor.instances).

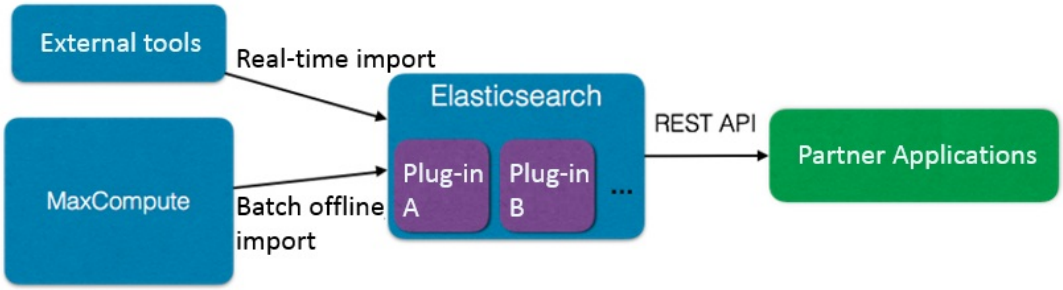
1.15. Elasticsearch on Maxcompute

1.15.1. Overview

Elasticsearch on MaxCompute is an enterprise-class full-text retrieval system developed by Alibaba Cloud for retrieving large volumes of data. It provides near-real-time (NRT) search performance for government agencies and enterprises. **Elasticsearch on MaxCompute** provides elastic full-text retrieval and supports native Elasticsearch APIs. Based on the APIs, it supports data import from heterogeneous data sources as well as cluster and service OAM. Based on the centralized scheduling and management capabilities of MaxCompute, Elasticsearch provides more efficient core services for retrieving large volumes of data.**Elasticsearch on MaxCompute** can also work with plugins available from the Elasticsearch open source community to provide a range of retrieval features.

You can import data to Elasticsearch on MaxCompute using external tools in real time or use MaxCompute to import offline data in batches. After indexing imported data, Elasticsearch on MaxCompute provides the retrieval service through the RESTful API. The following figure shows the usage of Elasticsearch on MaxCompute.

Elasticsearch on MaxCompute usage



1.15.2. Workflow

1.15.2.1. Overview

Elasticsearch on MaxCompute is based on the open source Elasticsearch. It can run the Elasticsearch service on MaxCompute clusters.

In the MaxCompute client, you can start and manage your Elasticsearch service as needed, including the number of nodes, disk space, memory size, and custom settings. The resources consumed by the Elasticsearch service are counted towards your MaxCompute instance quota.

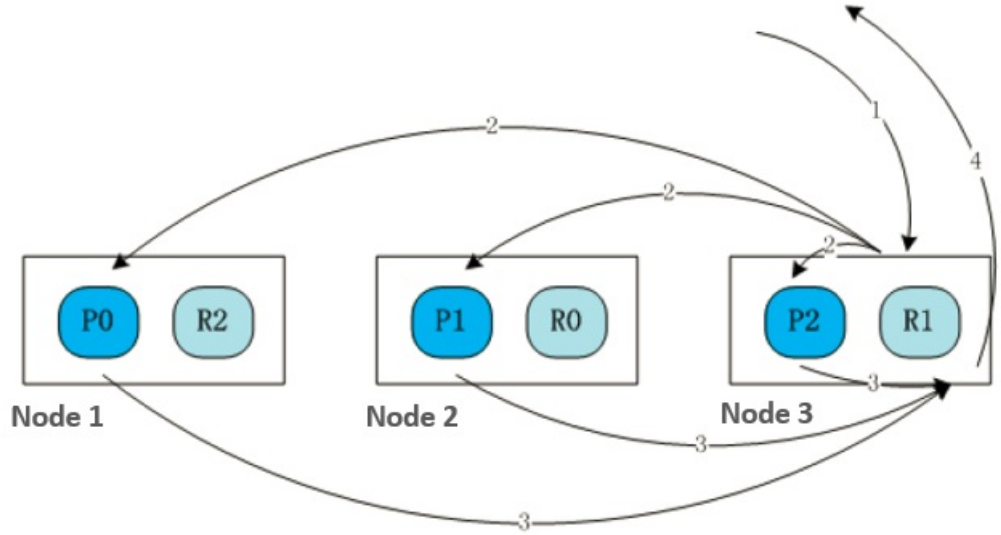
For the process of starting Elasticsearch, see the workflow chart in Typical practice of Elasticsearch.

The following topics describe the workflows of **Elasticsearch on MaxCompute** features.

1.15.2.2. Distributed retrieval workflow

The following figure shows the distributed retrieval workflow:

Distributed retrieval workflow



Each cluster consists of three nodes, as shown in the preceding figure. The index has three shards: P0, P1, and P2, which are distributed across three nodes. The shards work in 1:1 backup mode, so there are three replicas: R0, R1, and R2. The retrieval process is as follows:

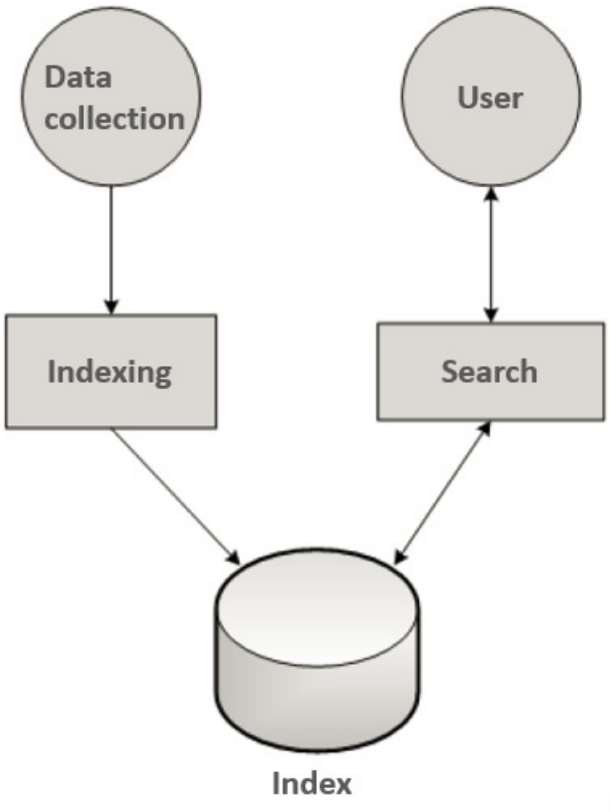
1. A user sends a retrieval request to node 3.
2. After receiving the request, node 3 sends a retrieval request (2) to P0, P1, and P2 based on the recorded index shard information.
3. The nodes where P0, P1, and P2 are located search for the requested information in the specified shards. Each node sends a retrieval result message (3) to node 3.
4. Node 3 combines the retrieval results from the other nodes, and returns a result to the user in an acknowledgment message (4).

Because multiple nodes perform data retrieval at the same time, the retrieval speed is improved. The performance of distributed retrieval increases with the number of nodes.

1.15.2.3. Full-text retrieval process

The following figure shows the full-text retrieval process.

Full-text retrieval process



The process is as follows:

1. The data collection module collects structured and unstructured data, converts the data into the field + value format, and submits the data to the index module.
2. The index module receives the data in the field + value format, performs segmentation and creates inverted indexes based on the predefined indexing method, and saves the indexes. The field type, indexing method, and segmentation rule are configured on the retrieval management page.
3. The search module receives and processes user requests. It parses the requests to obtain indexes, fields, and query statements. Then, the search module finds matching records from the inverted indexes.
4. The search module returns data that satisfies the user requirements, such as the sorting rule and quantity.

1.15.2.4. Authentication process

The authentication process is as follows:

1. **You try to log on to the Elasticsearch on MaxCompute** retrieval management or O&M platform. You are redirected to the authentication module. If you pass the authentication, you can access the platform. Otherwise, you are denied access to the platform.
2. The administrator can use the MaxCompute client to add Elasticsearch users and configure permissions for the users.
3. When you try to access the index library through an API, the system implements authentication. You can search for or operate data in the index library only after passing the authentication.

1.15.3. Quick start

Before you start an Elasticsearch cluster, make sure that you have determined the following information:

- **Node planning**: Determine the number of nodes that are required for each role in an Elasticsearch cluster. By default, an Elasticsearch cluster has two roles, master and data. Each role is deployed on three nodes. You can add nodes to the cluster at any time.
- **Resource planning**: Determine the vCPU, memory, and disk space resources that are required for each node. The resources configured for each node cannot be changed. By default, 8 GiB of memory and 20 GiB of disk space are allocated to each node.

  > **Note** Only 50% of the memory allocated to a data node is used for the JVM heap.

- **Elasticsearch configuration**: Determine the running configurations of nodes in an Elasticsearch cluster, such as the queue size of bulk requests, and support for cross-domain HTTP requests.

After you determine the preceding information, you can start your Elasticsearch cluster in the MaxCompute client.

The following example shows how to quickly start a small Elasticsearch cluster based on the default configurations. The name of the Elasticsearch cluster is es_first_cluster.

1. Download the MaxCompute client that supports Elasticsearch. Configure the AccessKey pair, project, and endpoint.
2. Start the client (odpscmd) and run the following command:

   ```
   server create es_first_cluster type elasticsearch_mdu;
   ```

   Wait for several minutes. If OK is returned, the Elasticsearch cluster is active. You must create an Elasticsearch user to access the cluster.
3. In odpscmd, create an Elasticsearch user with the permission of all_access.

   ```
   server execute es_first_cluster create user admin with password 123456|all_access;
   ```

   If OK is returned, the Elasticsearch user is created.
4. Access the started Elasticsearch cluster. Assume that the prj1 MaxCompute project is used to access the Elasticsearch cluster. Run the following command to return information about the Elasticsearch cluster:

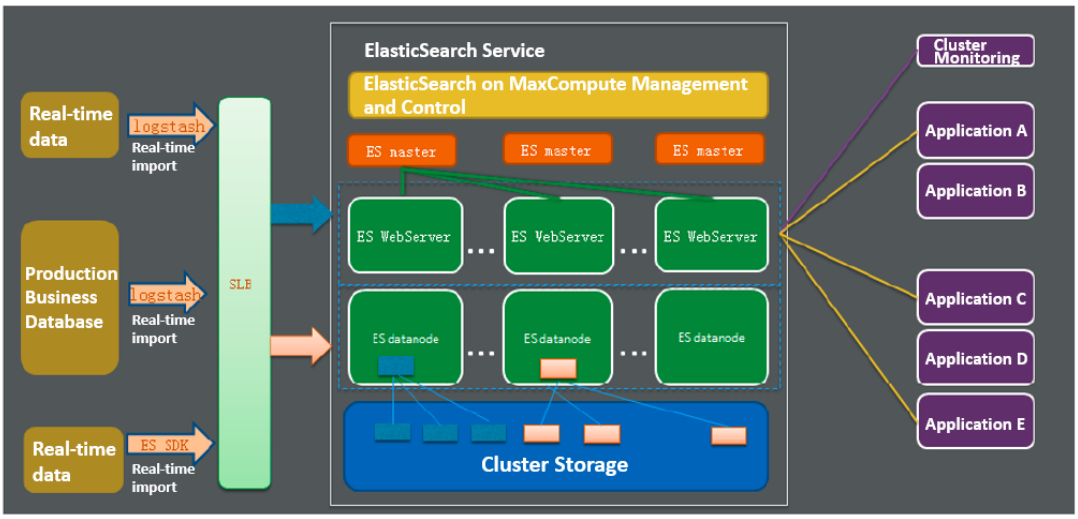
   ```
   curl -u admin:123456 http://search.aliyun.com:9200/prj1.es_first_cluster
   ```

> **Note** To delete the Elasticsearch cluster, run the `server delete es_first_cluster` command. If you run this command, the Elasticsearch cluster is **permanently deleted**, and data cannot be recovered.

1.15.4. Support for Elasticsearch applications

1.15.4.1. ElasticSearch typical practice

Typical practice



Elasticsearch on MaxCompute allows you to start a set of Elasticsearch services on MaxCompute clusters by submitting a job. The project does not modify the native Elasticsearch code. **Elasticsearch on MaxCompute** works in the same way as the native Elasticsearch cluster.

1.15.4.2. Elasticsearch on MaxCompute support for VPC

Alibaba Cloud Elasticsearch on MaxCompute is an enterprise-class full-text retrieval system for retrieving massive amounts of data. To comply with data isolation and security requirements, **Elasticsearch on MaxCompute** provides support for Virtual Private Cloud (VPC) networks so that you can apply access policies at VPC level. (Elasticsearch VPC limits).

Elasticsearch on MaxCompute supports VPC networks in the following model:

- Classic networks, VPC, and the Internet are isolated from each other. Users can access only the endpoints and virtual IP addresses (VIPs) on their networks.
- Projects without a whitelist of VPC IDs and IP addresses are accessible for users from valid domains over the three types of networks. A domain is valid only if its access request is acknowledged.
- When an ElasticSearch service instance is started in a MaxCompute project, they share the same VPCLIST, which is a whitelist of VPCs.
- Starting an ElasticSearch service instance occupies all resources by default. You must scale up the MaxCompute instance or scale down the ElasticSearch service instance if you start more ElasticSearch service instances.

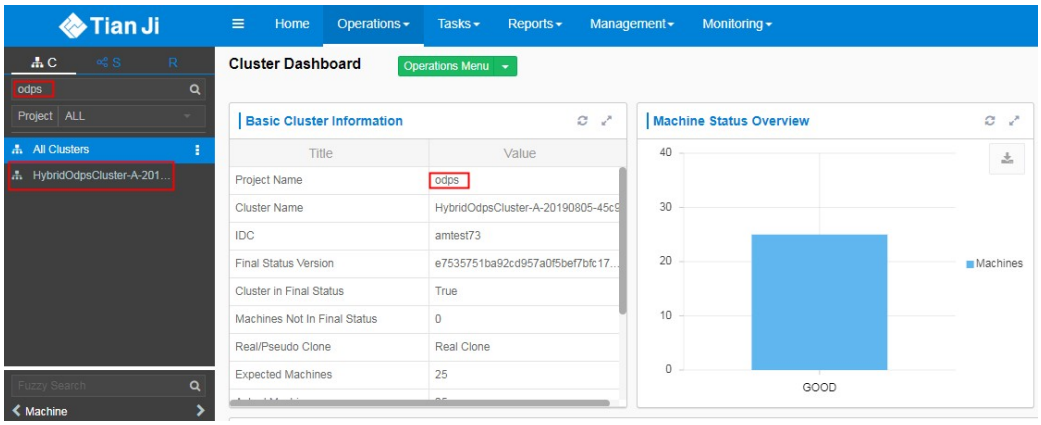
Here is a specific use example. Starting one ElasticSearch service instance for each project is taken as the default practice when a MaxCompute VPC is deployed. You can start your Elasticsearch instances for your projects, apply for domain names and VIPs, and perform VPC health check in the Elasticsearch frontend.

1.15.5. Special notes

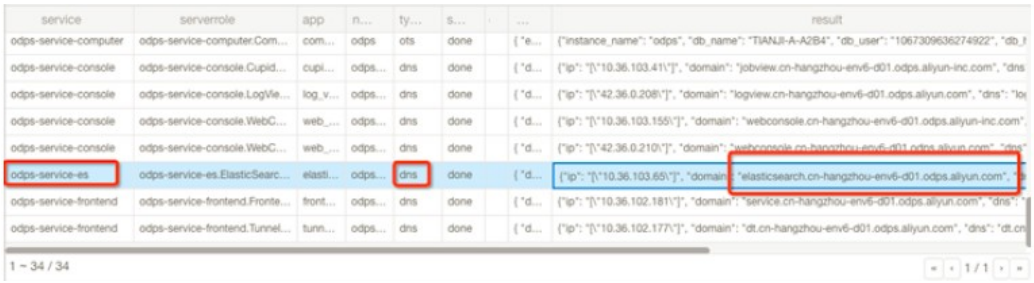
1.15.5.1. Find the Elasticsearch service domain name

1. Log on to the Apsara Infrastructure Management Framework console. Choose Operations >

Cluster Operations from the top navigation bar. In the left-side navigation pane, click the C tab and search for ODPS.



2. Navigate to the MaxCompute cluster resource usage page.
3. Find the Elasticsearch service domain name.



1.15.5.2. Import table data from MaxCompute to Elasticsearch

Before you can use **Elasticsearch on MaxCompute** to search for MaxCompute table data, you must import table data from MaxCompute to Elasticsearch clusters. MaxCompute MapReduce is designed to carry out the task of exporting data from MaxCompute to Elasticsearch. You can import data to Elasticsearch through simple configurations.

By utilizing the distributed dispatching capability of MaxCompute, you can easily control the concurrency. Besides, you can add the MapReduce job to the scheduled tasks on D2.

The data import process is as follows:

1. Download the JAR package of the MapReduce job.
2. Run the following command to add the JAR package to the MaxCompute resource files in the MaxCompute console:

```
add jar /PATH/TO/elasticsearch_output-1.0.0.jar
```

3. Create configuration file es_mr.conf in the following format for the MapReduce job:

```
<configuration>
 <property>
    <name>key1</name>
    <value>value1</value>
 </property>
 <property>
    <name>key2</name>
    <value>value2</value>
 </property>
</configuration>
```

4. Submit the MapReduce job in the MaxCompute console. Example:

```
jar -conf es_mr.conf -classpath /PATH/TO/elasticsearch_output-1.0.0.jar
      -resources elasticsearch_output-1.0.0.jar -Dworker_num=5
com.aliyun.odps.export.elasticsearch.mr.EsOutputJob <TABLE_NAME> [PARTITION_SPEC];
-- Five workers run concurrently to export data from <TABLE_NAME> [PARTITION_SPEC] to the Elas
ticsearch cluster.
```

The following table describes the parameters in the es_mr.conf file.

Parameters

Parameter	Example	Required	Default value	Description
es.resource	my_index/my_type	Yes	-	Index and type in the target Elasticsearch for imported data.
es.nodes	-	Yes	-	Elasticsearch endpoint.
es.nodes.client.only	true	No	false	Send data to client-only nodes only.
es.col.field.mapping	odps_col1:es_field1,odps_col3:es_field2	Yes	-	Mapping between the MaxCompute columns to be imported and the Elasticsearch fields.
es.batch.size.bytes	1 MB	No	1 MB	Batch data transfer size.
es.batch.size.entries	1000	No	1000	Number of data entries transferred each time.
es.net.http.auth.user	admin	No	-	Elasticsearch username.

Parameter	Example	Required	Default value	Description
es.net.http.auth.pass	123456	No	-	Elasticsearch password.
es.mapping.routing	field_routing	No	-	Routing field name, in the <CONSTANT> format for a constant.
es.mapping.id	field_id	No	-	id field of a document.

1.16. Flink on MaxCompute

In the current MaxCompute version, Flink on MaxCompute is only for trial use. The following demo describes only the trial features. For more information about new features, see later versions.

Demo:

1. Modify the configuration file.

   Decompress the Flink package, go to the configuration file directory *conf*, and make the following modifications to the flink-conf.yaml file:

   ```
   project_name: xxxx
   access_id: xxxx
   access_key: xxxx
   end_point: xxxx
   odps.cupid.distributedcache.mincopy: 1
   odps.cupid.proxy.domain.name:xxxx
   #odps.task.major.version:
   cupid_v2
   ```

   > **Note**
   > - The first four configuration items can be obtained from the odps_config.ini file under the */home/admin/odps/odps_tools/clt/conf/* directory.
   > - odps.cupid.proxy.domain.name specifies the domain name of odps_jobview_server_dns. The value is the domain name without sparkui.

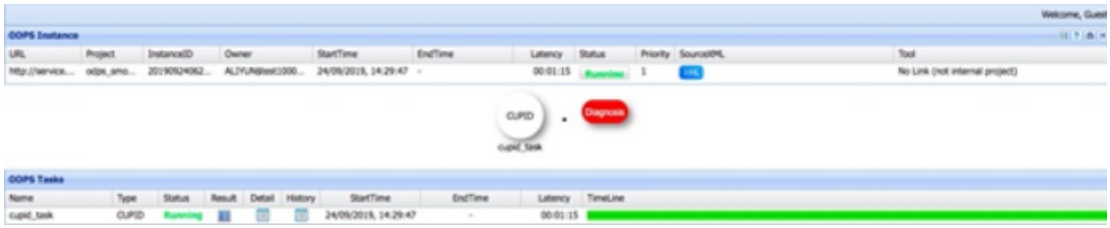
2. Start the program.

   Save flink-java-project-0.1.jar to the decompressed directory of Flink and run the following commands:

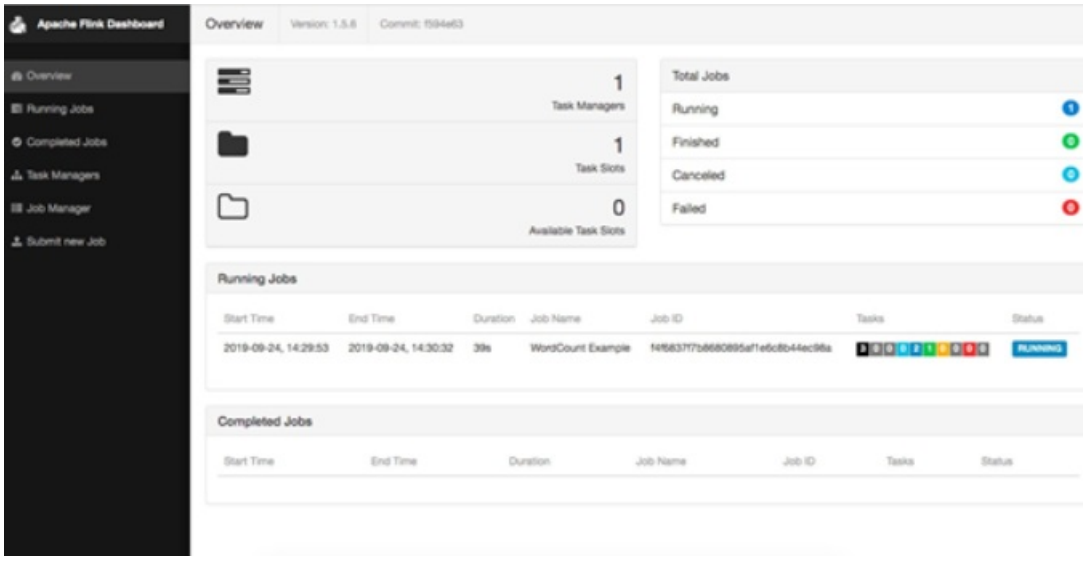
```
/bin/flink
run -c org.apache.flink.odps.WordCount -m yarn-cluster -yn 1
flink-java-project-0.1.jar --projectName odps_smoke_test --inputTable wc_in
--outputTable wc_out --sleepTime 100
```

3. Log on to Logview and Flink.

   Logview



   Flink



Note MaxCompute is compatible with Flink in terms of the following aspects:

- Group windows of Flink: The TUMBLE, HOP, and SESSION built-in functions are supported.
- Flink UDFs, UDAFs, and UDTFs: MaxCompute supports UDFs that run in Flink. You can use the `odps.sql.enable.flink.udf` flag to enable this feature.
- SQL syntax in Flink: The || operator is supported to connect strings. extract(dateunit from datetime) and substring(str from startIndex to endIndex) are supported.
- Flink HistoryServer: You can use Flink HistoryServer to view information about completed tasks.

1.17. Non-structured data access and processing (integrated computing scenarios)

1.17.1. Overview

MaxCompute SQL cannot directly process external data, such as non-structured data from Object Storage Service (OSS). This type of data must be imported to MaxCompute tables by using the relevant tools, which involves complex operations. The MaxCompute team introduced the non-structured data processing framework to the MaxCompute system architecture to simplify the processing of external data.

You can execute a DDL statement to create an external table in MaxCompute and associate the table with external data sources. This table can then act as an interface between MaxCompute and external data sources. External tables can be accessed like standard MaxCompute tables. You can fully use the computing capabilities of MaxCompute SQL to process external data.

MaxCompute allows you to create external tables to process data from the following data sources:

- Internal data sources: OSS, Tablestore, AnalyticDB, ApsaraDB for RDS, Alibaba Cloud HDFS, and TDDL
- External data sources: open source HDFS, MongoDB, and HBase

The subsequent topics describe various data sources.

> **Note** MaxCompute V2.0 supports multiple new data types. To use the new data types, you must add `set odps.sql.type.system.odps2=true;` before the SQL statement and commit them to enable the new data types. For ease of reading, sample code in subsequent topics uses new data types by default.

1.17.2. Internal data sources

1.17.2.1. OSS data source

1.17.2.1.1. Preface

As the core computing component of the Alibaba Cloud big data platform, MaxCompute provides powerful computing capabilities. It can schedule large amounts of nodes for parallel computing, and effectively manage failover and retry mechanisms in distributed computing. MaxCompute SQL implements different data processing logics through simple semantics. It is widely used within and outside Alibaba Group. It allows interoperability among different data sources and is significant for the integrated data ecology of Alibaba Cloud.

The following examples show how to access and process OSS unstructured data in MaxCompute.

1.17.2.1.2. Use the built-in extractor to read OSS data

1.17.2.1.2.1. Overview

You can use the MaxCompute built-in extractor to easily read OSS data in the specified format. You only need to create an external table as the source table for data query. For example, a CSV file is stored in OSS. The endpoint is oss-cn-shanghai-internal.aliyuncs.com, the bucket is oss-odps-test, and the file path is */demo/SampleData/CSV/AmbulanceData/vehicle.csv*. The following topics provide operation examples.

1.17.2.1.2.2. Create an external table

Execute the following statement to create an external table:

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_csv_external
(
vehicleId bigint,
recordId bigint,
patientId bigint,
calls bigint,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
STORED BY 'com.aliyun.odps.CsvStorageHandler'
LOCATION 'oss://<*your-id*>:<*your-secret-key*>@oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/Demo/SampleData/CSV/AmbulanceData/';
```

Note

- com.aliyun.odps.CsvStorageHandler is a built-in storage handler that is used to process CSV files. It defines how to read and write CSV files. You only need to specify this parameter. The logic is implemented by the system. To read and write TSV files, use com.aliyun.odps.TsvStorageHandler.
- LOCATION specifies an OSS directory. The system reads all files in the directory by default.
- An external table contains only related OSS directories. If you delete this table, the data in the directory specified by LOCATION is not deleted.

1.17.2.1.2.3. Query an external table

After an external table is created, you can use it the same way that you use a standard table.

The following content is included in the */demo/SampleData/CSV/AmbulanceData/vehicle.csv* file:

```
1,1,51,1,46.81006,-92.08174,9/14/2017 0:00,S
1,2,13,1,46.81006,-92.08174,9/14/2017 0:00,NE
1,3,48,1,46.81006,-92.08174,9/14/2017 0:00,NE
1,4,30,1,46.81006,-92.08174,9/14/2017 0:00,W
1,5,47,1,46.81006,-92.08174,9/14/2017 0:00,S
1,6,9,1,46.81006,-92.08174,9/14/2017 0:00,S
1,7,53,1,46.81006,-92.08174,9/14/2017 0:00,N
1,8,63,1,46.81006,-92.08174,9/14/2017 0:00,SW
1,9,4,1,46.81006,-92.08174,9/14/2017 0:00,NE
1,10,31,1,46.81006,-92.08174,9/14/2017 0:00,N
```

Execute the following statement to submit a job, which calls a built-in CSV extractor to read data from OSS:

```
SELECT recordId, patientId, direction
FROM ambulance_data_csv_external
WHERE patientId > 25;
```

> **Note**
>
> - An external table can only be managed by using MaxCompute SQL, not MaxCompute MapReduce.
> - To obtain data over HTTPS at the underlying layer, add the `set odps.sql.unstructured.data.oss.use.https=true;` flag before an SQL statement. Then, commit them for execution.

The following result is returned:

```
+------------+------------+-----------+
| recordId | patientId | direction |
+------------+------------+-----------+
| 1 | 51 | S |
| 3 | 48 | NE |
| 4 | 30 | W |
| 5 | 47 | S |
| 7 | 53 | N |
| 8 | 63 | SW |
| 10 | 31 | N |
+------------+------------+-----------+
```

> **Note** The system provides built-in CsvStorageHandler, TsvStorageHandler, and TextStorageHandler.

1.17.2.1.2.4. MSCK REPAIR TABLE

The **MSCK REPAIR TABLE** statement can be used to add partitions to external tables. Syntax:

```
MSCK [REPAIR] TABLE external_table_name [ADD PARTITIONS];
```

To add partitions, perform the following steps:

1. Import data to Object Storage Service (OSS). The storage path must be in the following format: `oss://xxx/table-location/ptname1=ptvalue1/ptname2=ptvalue2/xxx` .
2. Create an external table and specify the structures of the ptname1 and ptname2 partitions.
3. Execute the `msck repair table external_table_name [add partitions]` statement. MaxCompute SQL automatically parses the OSS directory structure, identifies partitions, and adds partitions to the external table.

Sample code:

```
CREATE EXTERNAL TABLE orc_pt_v0
(
name string
)
partitioned by (pt bigint)
STORED AS textfile
location 'oss://xxx/odps-ext-reg-perf/orc-pt-v0';msck repair table orc_pt_v0 add partitions;
-- In this case, the MSCK statement is equivalent to the following three statements:
alter table orc_pt_v0 add partition (pt=1);
alter table orc_pt_v0 add partition (pt=10);
alter table orc_pt_v0 add partition (pt=100);
```

1.17.2.1.3. Custom extractors

1.17.2.1.3.1. Overview

If OSS data is in a complicated format that cannot be processed by the built-in extractor, you must customize an extractor to read data from OSS files. For example, a text file is stored in OSS. The file is not in the CSV format and the columns of records are separated by vertical bars (|). The file path is */demo/SampleData/CustomTxt/AmbulanceData/vehicle.csv*. The following topics provide operation examples.

1.17.2.1.3.2. Define StorageHandler

You can customize the data parsing logic. StorageHandler is the unified entrance of your custom logic. You can specify the types of custom extractors and outputers. StorageHandler provides only a simple definition. For example, you can implement SpeicalTextStorageHandler:

```
package com.aliyun.odps.udf.example.text;
public class SpeicalTextStorageHandler extends OdpsStorageHandler {
@Override
public Class&lt;? extends Extractor&gt; getExtractorClass() {
return TextExtractor.class;
}
@Override
public Class&lt;? extends Outputer> getOutputerClass() {
return TextOutputer.class;
}
}
```

> **Note** Note that TextStorageHandler that is built in MaxCompute can process the data format in the preceding example (text separated by vertical bars (|)). This example is provided only to show you how to use the SDK to customize a StorageHandler (especially extractor) for processing uncommonly structured data.

1.17.2.1.3.3. Define an extractor

In the following example, TextExtractor is used to extract records from a text file, where the delimiter is imported as a parameter. TextExtractor can be used for all text files of the similar format.

```
/**
* Text extractor that extract schematized records from formatted plain-
text(csv, tsv etc.)
**/
public class TextExtractor extends Extractor {
private InputStreamSet inputs;
private String columnDelimiter;
private DataAttributes attributes;
private BufferedReader currentReader;
private boolean firstRead = true;
public TextExtractor() {
// Default to ",", this can be overwritten if a specific delimiter is
provided (via DataAttributes)
this.columnDelimiter = ",";
}
// No particular usage for execution context in this example
```

```
@Override
public void setup(ExecutionContext ctx, InputStreamSet inputs,
DataAttributes attributes) {
this.inputs = inputs;
-- inputs specifies an InputStreamSet. An InputStream is returned each time next() is called. This Input
Stream can read all the content of an OSS file.
this.attributes = attributes;
// Check if "delimiter" attribute is supplied via SQL query
String columnDelimiter = this.attributes.getValueByKey("delimiter");
-- The delimiter can be used as a parameter in DDL statements.
if ( columnDelimiter ! = null)
{
this.columnDelimiter = columnDelimiter;
}
// note: more properties can be inited from attributes if needed
}
@Override
public Record extract() throws IOException {
String line = readNextLine();
if (line == null) {
return null;
-- If NULL is returned, all records in the table have been read.
}
return textLineToRecord(line);
-- textLineToRecord splits a row into multiple columns using the delimiter. For the implementation proc
ess, see Complete TextExtractor implementation.
-- extractor() returns a record that is extracted from OSS data.
}
@Override
public void close(){
// no-op
}
}
```

1.17.2.1.3.4. Compile and package code

You can compile and package Java code, and run the following command to upload it to MaxCompute. The procedure is the same as that for a normal Java UDF.

add jar odps-udf-example.jar;

1.17.2.1.3.5. Create an external table

After you upload a JAR package, you need to run the following command to create an external table. This command is similar to the one that you run before using a built-in extractor. The difference is that this command uses a custom StorageHandler.

```
CREATE EXTERNAL TABLE IF NOT EXISTS ambulance_data_txt_external
(
vehicleId int,
recordId int,
patientId int,
calls int,
locationLatitute double,
locationLongtitue double,
recordTime string,
direction string
)
STORED BY 'com.aliyun.odps.udf.example.text.SpeicalTextStorageHandler'
-- STORED BY specifies the class name of a custom StorageHandler.
WITH SERDEPROPERTIES('delimiter'='|')
-- SERDEPROPERITES can be used to specify parameters. These parameters are transferred to an extra
ctor through DataAttributes.
LOCATION 'oss://<*your-id*>:<*your-secret-key*>@oss-cn-shanghai-internal.aliyuncs.com/oss-odps-t
est/Demo/SampleData/CustomTxt/AmbulanceData/'
USING 'odps-udf-example.jar';
-- Specify the JAR package where the class definition is located.
```

1.17.2.1.3.6. Query an external table

The content of */demo/SampleData/CustomTxt/AmbulanceData/vehicle.csv* is as follows:

```
1|1|51|1|46.81006|-92.08174|9/14/2017 0:00|S
1|2|13|1|46.81006|-92.08174|9/14/2017 0:00|NE
1|3|48|1|46.81006|-92.08174|9/14/2017 0:00|NE
1|4|30|1|46.81006|-92.08174|9/14/2017 0:00|W
1|5|47|1|46.81006|-92.08174|9/14/2017 0:00|S
1|6|9|1|46.81006|-92.08174|9/14/2017 0:00|S
1|7|53|1|46.81006|-92.08174|9/14/2017 0:00|N
1|8|63|1|46.81006|-92.08174|9/14/2017 0:00|SW
1|9|4|1|46.81006|-92.08174|9/14/2017 0:00|NE
1|10|31|1|46.81006|-92.08174|9/14/2017 0:00|N
```

Run the following command to submit a job, which calls a custom extractor to read data from OSS:

```
SELECT recordId, patientId, direction
FROM ambulance_data_txt_external
WHERE patientId > 25;
```

Command output:

```
+------------+------------+-----------+
| recordId | patientId | direction |
+------------+------------+-----------+
| 1 | 51 | S |
| 3 | 48 | NE |
| 4 | 30 | W |
| 5 | 47 | S |
| 7 | 53 | N |
| 8 | 63 | SW |
| 10 | 31 | N |
+------------+------------+-----------+
```

1.17.2.1.4. Advanced usage

1.17.2.1.4.1. Use a custom extractor to read external unstructured data

The preceding topic describes how to use built-in and custom extractors to process text files such as .csv files that are stored in OSS. This topic describes how to use UDF extractors to process non-text files in OSS.

The following example shows how to process audio files (.wav files) in OSS.

1. Customize the SpeechSentenceSnrExtractor main logic. Use the SETUP API to read parameters, initialize the parameters, and import the audio processing model (by using the resource function).

```
public SpeechSentenceSnrExtractor(){
this.utteranceLabels = new HashMap&lt;String, UtteranceLabel&gt;();
}
@Override
public void setup(ExecutionContext ctx, InputStreamSet inputs,
DataAttributes attributes){
this.inputs = inputs;
this.attributes = attributes;
this.mlfFileName = this.attributes.getValueByKey(MLF_FILE_ATTRIBUTE_KEY);
```

```
this.mlfFileName = this.attributes.getValueByKey(MLF_FILE_ATTRIBUTE_KEY);
String sampleRateInKHzStr =
this.attributes.getValueByKey(SPEECH_SAMPLE_RATE_KEY);
this.sampleRateInKHz = Double.parseDouble(sampleRateInKHzStr);
try {
// read the speech model file from resource and load the model into
memory
BufferedInputStream inputStream =
ctx.readResourceFileAsStream(mlfFileName);
loadMlfLabelsFromResource(inputStream);
inputStream.close();
} catch (IOException e) {
throw new RuntimeException("reading model from mlf failed with exception
" + e.getMessage());
}
}
@Override
public Record extract() throws IOException {
SourceInputStream inputStream = inputs.next();
if (inputStream == null){
return null;
}
// process one wav file to extract one output record [snr, id]
String fileName = inputStream.getFileName();
fileName = fileName.substring(fileName.lastIndexOf('/') + 1);
logger.info("Processing wav file " + fileName);
// infer id from speech file name
String id = fileName.substring(0, fileName.lastIndexOf('.'));
// read speech file into memory buffer
long fileSize = inputStream.getFileSize();
byte[] buffer = new byte[(int)fileSize];
int readSize = inputStream.readToEnd(buffer);
inputStream.close();
// compute the avg sentence snr from speech file
double snr = computeSnr(id, buffer, readSize);
// construct output record [snr, id]
Column[] outputColumns = this.attributes.getRecordColumns();
ArrayRecord record = new ArrayRecord(outputColumns);
record.setDouble(0, snr);
record.setString(1, id);
return record;
```

```
}
private void loadMlfLabelsFromResource(BufferedInputStream fileInputStream)
throws IOException {
// loading MLF label from resource, skipped here
}
// compute the snr of the speech sentence, assuming the input buffer
contains the entire content of a wav file
private double computeSnr(String id, byte[] buffer, int validBufferLen){
// computing the snr value for the wav file (supplied as byte buffer
array), skipped here
}
```

> **Note** The Extractor() API implements the reading and processing logic of audio files. It calculates the signal-to-noise ratio (SNR) of the read data based on the audio processing model and writes the result to a record in [snr, id] format.

2. Run the following commands to create an external table:

```
CREATE EXTERNAL TABLE IF NOT EXISTS speech_sentence_snr_external
(
sentence_snr double,
id string
)
STORED BY 'com.aliyun.odps.udf.example.speech.SpeechStorageHandler'
WITH SERDEPROPERTIES (
'mlfFileName'='sm_random_5_utterance.text.label' ,
'speechSampleRateInKHz' = '16'
)
LOCATION 'oss://<*your-id*>:<*your-secret-key*>@oss-cn-shanghai-internal.aliyuncs.com/oss-od
ps-test/dev/SpeechSentenceTest/'
USING 'odps-udf-example.jar,sm_random_5_utterance.text.label';
```

3. Run the following commands to read data from OSS:

```
SELECT sentence_snr, id
FROM speech_sentence_snr_external
WHERE sentence_snr > 10.0;
```

4. The command output is as follows:

```
------------------------------------------------------------
| sentence_snr | id |
------------------------------------------------------------
| 34.4703 | J310209090013_H02_K03_042 |
------------------------------------------------------------
| 31.3905 | tsh148_seg_2_3013_3_6_48_80bd359827e24dd7_0 |
------------------------------------------------------------
| 35.4774 | tsh148_seg_3013_1_31_11_9d7c87aef9f3e559_0 |
------------------------------------------------------------
| 16.0462 | tsh148_seg_3013_2_29_49_f4cb0990a6b4060c_0 |
------------------------------------------------------------
| 14.5568 | tsh_148_3013_5_13_47_3d5008d792408f81_0 |
------------------------------------------------------------
```

Note By using the UDF extractor, you can run SQL statements to process multiple audio files in OSS in a distributed manner. Similarly, you can use the large-scale computation capabilities of MaxCompute to process unstructured data such as images and videos.

1.17.2.1.5. Data partitions

1.17.2.1.5.1. Overview

In the preceding topic, LOCATION is used to specify an OSS directory, which is associated with an external table. MaxCompute reads all data in the directory, including all files in the subdirectories. When there is a large volume of data in the directory, a full-text scan will cause extra I/O operations and processing time. There are two solutions:

- Reducing the data volume: You need to properly plan data storage addresses. Create multiple external tables for data from different parts, with the LOCATION of each external table pointing to a subset of data.
- Partitioning data: The external table, like an internal table, supports partitioning. You can create partitions to facilitate data management.

The following topics describe the partition feature of external tables.

1.17.2.1.5.2. Standard organization method and path format of partition data in OSS

Unlike the data in internal tables, data stored in external storage (such as OSS) cannot be managed in MaxCompute. If you need to use the partitioned table feature of MaxCompute, make sure that the data file paths in OSS are in the following format:

```
partitionKey1=value1\partitionKey2=value2\...
```

Example:

1. Your daily log files are stored in OSS, and you want to access some of the data from MaxCompute on a daily basis. If the log files are in the CVS format or a similar custom format, you can execute the following statement to create a partitioned external table:

```
CREATE EXTERNAL TABLE log_table_external (
click STRING,
ip STRING,
url STRING,
)
PARTITIONED BY (
year STRING,
month STRING,
day STRING
)
STORED BY 'com.aliyun.odps.CsvStorageHandler'
LOCATION 'oss://<ak_id>:<ak_key>@oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/log_d
ata/';
```

> **Note** In the preceding example, the PARTITIONED BY clause is used to specify a partitioned external table. The partition keys are year, month, and day.

2. For the partitions to take effect, you must specify the OSS storage directory in the format shown in the preceding example. An example of a valid directory layout is as follows:

```
osscmd ls oss://oss-odps-test/log_data/
2017-09-14 08:03:35 128MB Standard oss://oss-odps-
test/log_data/year=2017/month=06/day=01/logfile
2017-09-14 08:04:12 127MB Standard oss://oss-odps-
test/log_data/year=2017/month=06/day=01/logfile. 1
2017-09-14 08:05:02 118MB Standard oss://oss-odps-
test/log_data/year=2017/month=06/day=02/logfile
2017-09-14 08:06:45 123MB Standard oss://oss-odps-
test/log_data/year=2017/month=07/day=10/logfile
2017-09-14 08:07:11 115MB Standard oss://oss-odps-
test/log_data/year=2017/month=08/day=08/logfile
...
```

> **Note** If you have uploaded offline data to OSS by using osscmd or other OSS tools, you can define the data path format. To ensure the partitioned external table feature operates normally, we recommend that the path where data is stored to is in the format specified in the previous example.

3. Then, you can execute the ALTER TABLE ADD PARTITION statement to import the partition information to MaxCompute. An example of the DDL statement is as follows:

```
ALTER TABLE log_table_external ADD PARTITION (year = '2017', month = '06', day = '01')
ALTER TABLE log_table_external ADD PARTITION (year = '2017', month = '06', day = '02')
ALTER TABLE log_table_external ADD PARTITION (year = '2017', month = '07', day = '10')
ALTER TABLE log_table_external ADD PARTITION (year = '2017', month = '08', day = '08')
...
```

4. When the data is ready and the partition information has been imported to MaxCompute, you can execute SQL statements to operate the partitions in the external table in OSS.
   - Execute the following statement to count the number of unique IP addresses in the log dated June 1, 2017:

     ```
     SELECT count(distinct(ip)) FROM log_table_external
     WHERE year = '2017' AND month = '06' AND day = '01';
     ```

     Note In the log_table_external directory that corresponds to an external table, only files in the *log_data/year=2017/month=06/day=01* subdirectory (logfile and To prevents unnecessary I/O operations, a full scan of the *log_data/* directory is not performed.

   - Similarly, you can execute the following statement to analyze data for the second half of 2017:

     ```
     SELECT count(distinct(ip)) FROM log_table_external
     WHERE year = '2017' AND month > '06';
     ```

     Note In this case, only the logs for the second half of 2017 stored in OSS are accessed.

1.17.2.1.5.3. Custom path of partition data in OSS

If you have historical data stored in OSS paths that are not in the partitionKey1=value1\partitionKey2=value2\... format, you can still access the data by using the MaxCompute partition feature. MaxCompute provides a way to import partitions through a custom path.

Example:

1. The data path only contains partition values (without partition keys). An example of the path layout is as follows:

```
osscmd ls oss://oss-odps-test/log_data_customized/
2017-09-14 08:03:35 128MB Standard oss://oss-odps-
test/log_data_customized/2017/06/01/logfile
2017-09-14 08:04:12 127MB Standard oss://oss-odps-
test/log_data_customized/2017/06/01/logfile. 1
2017-09-14 08:05:02 118MB Standard oss://oss-odps-
test/log_data_customized/2017/06/02/logfile
2017-09-14 08:06:45 123MB Standard oss://oss-odps-
test/log_data_customized/2017/07/10/logfile
2017-09-14 08:07:11 115MB Standard oss://oss-odps-
test/log_data_customized/2017/08/08/logfile
...
```

2. You can run the following statement to bind subdirectories to different partitions:

```
ALTER TABLE log_table_external ADD PARTITION (year = '2017', month = '06', day = '01')
LOCATION 'oss://<ak_id>:<ak_key>@oss-cn-shanghai-internal.aliyuncs.com/oss-odps-test/log_d
ata_customized/2017/06/01/';
```

> **Note** The ADD PARTITION and LOCATION clauses are specified in the preceding example to bind the partitions to data paths. Even if the data storage path is not in the partitionKey1=value1\partitionKey2=value2\... format, you can still access the partition data in the subdirectory.

1.17.2.1.5.4. Access fully-customized non-partitioned data subsets

In certain situations, you might need to access a file subset in an OSS path, but files in this subset do not have any obvious regularity in terms of directory layout. The unstructured data processing framework of MaxCompute is able to handle this situation, but will not be discussed in this topic.

If you require advanced operations such as this, contact the MaxCompute technical team for support.

1.17.2.1.6. Output OSS data

1.17.2.1.6.1. Create an external table

To write data to OSS, you need to run the CREATE EXTERNAL TABLE statement to create an external table first. The process is the same as that of reading data from OSS. After the external table is created, you can run MaxCompute SQL statements such as INSERT INTO and INSERT OVERWRITE to write data to OSS. In the following example, the built-in TsvStorageHandler is used.

```
DROP TABLE IF EXISTS tpch_lineitem_tsv_external;
CREATE EXTERNAL TABLE IF NOT EXISTS tpch_lineitem_tsv_external
(
orderkey BIGINT,
suppkey BIGINT,
discount DOUBLE,
tax DOUBLE,
shipdate STRING,
linestatus STRING,
shipmode STRING,
comment STRING
)
STORED BY 'com.aliyun.odps.TsvStorageHandler'
LOCATION 'oss://<AK_id>:<AK_secret>@oss-cn-hangzhou-zmf.aliyuncs.com/oss-odps-test/tsv_output
_folder/';
```

Note

The preceding DDL statement creates an external table named tpch_lineitem_tsv_external, and associates two external data dimensions with this external table.

- Data storage medium: LOCATION associates an OSS address with the external table. This address will be used to read or write data from or to the external table.
- Data storage format: StorageHandler is used to define the data access mode. In this example, MaxCompute built-in com.aliyun.odps.TsvStorageHandler is used to read or write data from or to TSV files. You can also use the MaxCompute SDK to define StorageHandlers.

1.17.2.1.6.2. Write data to a TSV text file by using an INSERT statement on an external table

After you associate a file in OSS with an external table, you can run a standard SQL INSERT OVERWRITE/INSERT INTO statement on the external table to write data to the OSS file. The source data can be either data stored in a MaxCompute internal table or external data that is imported to MaxCompute through an external table.

> **Note**
>
> - MaxCompute internal table: You can run an INSERT statement on an external table to write data from a MaxCompute internal table to an external storage medium.
> - External data imported to MaxCompute through an external table: You can import external data to MaxCompute through an external table, use the data for computations, and then export the results to an external address or storage medium. For example, import Table Store data to MaxCompute and then export the data to OSS.

The following example assumes that you have a MaxCompute internal table named tpch_lineitem and want to export some of the data to OSS in the TSV format. After you create an external table, run the following INSERT OVERWRITE statements to export data:

```
INSERT OVERWRITE TABLE tpch_lineitem_tsv_external
SELECT l_orderkey, l_suppkey, l_discount, l_tax, l_shipdate, l_linestatus,
l_shipmode, l_comment
FROM tpch_lineitem
WHERE l_discount = 0.07 and l_tax = 0.01;
```

The preceding example selects eight columns from the rows in tcph_lineitem table that satisfy the conditions l_discount = 0.07 and l_tax = 0.01 and writes it to tpch_lineitem_tsv_external in OSS in the TSV format. After this operation is complete, you can view the corresponding TSV data file in OSS.

> **Notice**
>
> Data exported from MaxCompute to OSS is stored in a special file structure.
>
> - When you run INSERT INTO/OVERWRITE statements on an OSS address, all data is exported to the .odps folder at the specified LOCATION.
> - The .meta file in the .odps folder is an extra macro data file written by MaxCompute to record valid data in the current folder. Typically, if the INSERT operation is successful, all the data in the current folder is valid. You are only required to parse the macro data if a job fails.
> - If a job fails or is terminated, perform the INSERT OVERWRITE operation again until it is complete. This prevents parsing of the .meta file.
> - If you need to parse the .meta file, contact Alibaba Cloud technical team for support.

The number of files that are generated during MaxCompute built-in TSV/CSV processing is equal to the number of concurrent SQL stages. You can use the flexible semantics and configurations of MaxCompute to limit the number of generated files. In the preceding example, if you need to force a TSV file to be generated, you can append DISTRIBUTE BY l_discount to the INSERT OVERWRITE operation. Then, a reduce stage with only one reducer is added at last so that only one TSV file is output.

1.17.2.1.6.3. Write data to an unstructured file by using an INSERT statement on an external table

MaxCompute also provides Outputer APIs for data output. You can use the APIs to write user data to a custom unstructured data file through OutputStream. Further details are not covered in this topic.

If you have this requirement, contact the MaxCompute technical team for support.

1.17.2.1.6.4. Migrate data between different storage media with MaxCompute

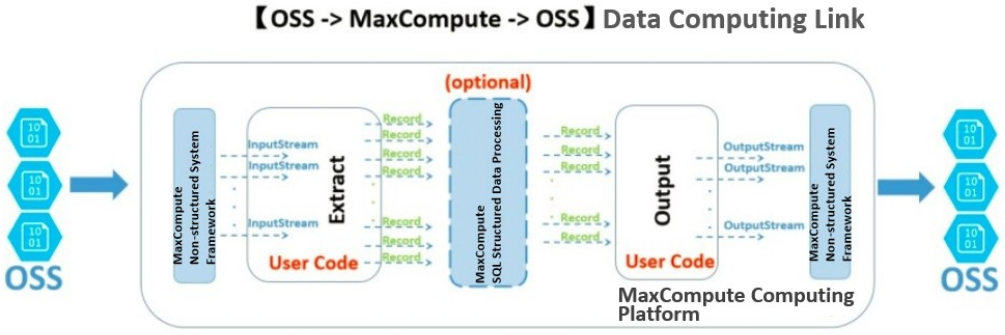
External tables act as an interface between MaxCompute and external storage media. External tables can be used to read or write data from or to various external storage media such as OSS and Table Store. Based on the external table feature, various data computing and storage links can be established. For example,

1. MaxCompute reads the OSS data associated with External Table A, and performs complicated computations. MaxCompute then outputs the results to the OSS address associated with External Table B.
2. MaxCompute reads the Table Store data associated with External Table A, and performs complicated computations. MaxCompute then outputs the results to the OSS address associated with External Table B.

> **Note** The preceding examples and data sources are scenarios with MaxCompute tables. The only difference is that the SELECT statement originates from an external table instead of a MaxCompute table.

Example:

By using MaxCompute as a central computing platform, you can import data from an OSS instance, and export that data to a different OSS instance (in a different location, or a different OSS account), as shown in the following figure.



From a data flow and processing logic standpoint, the unstructured data processing framework can be considered as a coupled data ingress and egress at both ends of the MaxCompute platform.

1. The external data (from an OSS instance) is converted based on the unstructured

framework, and provided to the UDF API in the form of a Java InputStream class. The UDF extract logic is only required to read, parse, transform, and compute the data from the InputStream class, and return the data in the Record format used by MaxCompute.

2. Part of the returned records are used in the SQL logical operations on MaxCompute. These operations utilize the powerful SQL computation engine built into MaxCompute, and may generate new records.
3. The computed records are transferred to the UDF Output logic for further computation. Finally, the required information is extracted from the records, output through OutputStream, and written to the OSS instance.

> **Notice** You can perform any combination of the preceding steps based on your needs.

1.17.2.1.7. STS mode authorization for OSS

When you create an external table, the Location-based OSS access account supports plaintext input of the AccessKey ID and AccessKey secret, but the account information may be exposed. To prevent the leakage of account information, MaxCompute provides a more secure way to access OSS.

MaxCompute combines RAM and Security Token Service (STS) of Alibaba Cloud to resolve security issues of accounts. You can grant permissions in two ways:

- If the owners of MaxCompute and OSS use the same Alibaba Cloud account, you can authorize access to OSS with one click in the RAM console.
- Custom authorization is supported.
  i. You can log on to the RAM console and authorize access to OSS.

     Create a role named AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser and set the policy content as follows:

```
-- If the owners of MaxCompute and OSS use the same account:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
-- If the owners of MaxCompute and OSS use different accounts:
{
"Statement": [
{
 "Action": "sts:AssumeRole",
 "Effect": "Allow",
 "Principal": {
   "Service": [
     "ID of the Alibaba Cloud account used by the owner of MaxCompute@odps.aliyuncs.com"
   ]
 }
}
],
"Version": "1"
}
```

ii. Set the AliyunODPSRolePolicy permission, which is the necessary permission for the role to access OSS.

```
{
"Version": "1",
"Statement": [
{
 "Action": [
   "oss:ListBuckets",
   "oss:GetObject",
   "oss:ListObjects",
   "oss:PutObject",
   "oss:DeleteObject",
   "oss:AbortMultipartUpload",
   "oss:ListParts"
 ],
 "Resource": "*",
 "Effect": "Allow"
}
]
}
--You can also add other permissions as required.
```

iii. Grant the AliyunODPSRolePolicy permission to the role.

> **Note** After authorization is complete, view the role details to obtain the RAM information of this role. You need to specify the RAM information when you create an OSS external table later.

1.17.2.2. Table Store data source

1.17.2.2.1. Preface

As the core computing component of the Alibaba Cloud big data platform, MaxCompute meets most distributed computing requirements within and outside Alibaba Group. As the entry of distributed data processing, MaxCompute SQL provides powerful support for quick processing and storage of large volumes (exabytes) of offline data. With the continuous expansion of big data business, many new data usage scenarios emerge. To adapt to the new scenarios, the MaxCompute computing framework is constantly evolving. Its powerful computation capabilities, originally designed to process internal data in special formats, have expanded to process external data sources in various formats. This topic describes in detail how to import data from Table Store to MaxCompute, implementing seamless interoperability between data sources.

Compared with traditional databases, NoSQL KV Store supports flexible schema, scalability and real-time response for applications such as online service. Alibaba Cloud Table Store is a large-scale NoSQL data storage service based on the Apsara system. It supports storage and real-time access of massive KV data. Table Store is widely used by all business units in Alibaba Group and the Alibaba Cloud ecosystem. In particular, Table Store features, such as row-level real-time update and override writing, are a supplement to the append-only operation of MaxCompute tables. As a storage-oriented service, Table Store does not provide sufficient computing capabilities to process large amounts of data concurrently. This makes it important to enable data interoperability between MaxCompute and Table Store.

The following examples show how to access and process Table Store data in MaxCompute.

1.17.2.2.2. MaxCompute reads and computes data in Table Store

1.17.2.2.2.1. Prerequisites and assumptions

This document assumes that you have basic knowledge of Table Store operations. If you are not familiar with Table Store or KV tables, we recommend that you first familiarize yourself with the basic concepts of Table Store (such as primary keys, partition keys, and attribute columns) before reading this topic.

1.17.2.2.2.2. Create an external table

External tables can act as interfaces between MaxCompute and Table Store: You can run a DDL statement (CREATE EXTERNAL TABLE) to import a table description in Table Store to the MaxCompute meta system. Then, you can process data in the Table Store table in the same way you process data in a MaxCompute table.

Example:

```
DROP TABLE IF EXISTS ots_table_external;
CREATE EXTERNAL TABLE IF NOT EXISTS ots_table_external
(
odps_orderkey bigint,
odps_orderdate string,
odps_custkey bigint,
odps_orderstatus string,
odps_totalprice double
)
STORED BY 'com.aliyun.odps.TableStoreStorageHandler'
-- com.aliyun.odps.TableStoreStorageHandler is a MaxCompute built-in StorageHandler for processing Table Store data. It defines the interaction between MaxCompute and Table Store. The relevant logic is implemented by MaxCompute.
WITH SERDEPROPERTIES (
-- SERDEPROPERITES is an API that provides parameter options. Two options must be specified for TableStoreStorageHandler: tablestore.columns.mapping and tablestore.table.name.
'tablestore.columns.mapping'=':o_orderkey, :o_orderdate, o_custkey,
o_orderstatus,o_totalprice',
-- tablestore.columns.mapping: This option is required. It describes the columns of Table Store tables that are accessed by MaxCompute, including primary key and attribute columns. Column names starting with a colon (:) are primary key columns in Table Store tables. In this example, :o_orderkey and :o_orderdate are primary key columns. The other column names specified are attribute columns. Table Store supports up to four primary keys of the bigint or string type. The first primary key is the partition key. When you specify a mapping, you must provide all primary key columns of the specified Table Store table. You do not have to specify all the attribute columns, only those accessed by MaxCompute.
'tablestore.table.name'='ots_tpch_orders'
-- tablestore.table.name: This option is required. It describes the names of Table Store tables that are accessed by MaxCompute. If you specify an invalid (nonexistent) Table Store table name, an error is returned and MaxCompute does not create a Table Store table with this name.
)
LOCATION 'tablestore://<*your AK id*>:<*your AK secret key*>@odps-ots-dev.cn-
hangzhou.ots.aliyuncs.com';
-- The LOCATION clause specifies the Table Store information, including the instance name and endpoint.
```

Note The preceding example maps a Table Store table to a MaxCompute external table. The subsequent operations on the Table Store table can be performed through the external table.

1.17.2.2.2.3. Access Table Store data through an external table

After you follow the preceding example to create an external table, Table Store data is imported to MaxCompute. Then, you can access Table Store data by using MaxCompute SQL statements.

Example:

```
SELECT odps_orderkey, odps_orderdate, SUM(odps_totalprice) AS totalprice
FROM ots_table_external
WHERE odps_orderkey > 5000 AND odps_orderdate >20170725 AND odps_orderdate <20170910
GROUP BY odps_orderkey, odps_orderdate
HAVING totalprice> 2000;
```

> **Note** This example uses common MaxCompute SQL statements. Table Store access details are processed internally by MaxCompute.

If you need to use a copy of data for multiple computations, you can import the data from Table Store to a MaxCompute table (internal). This is more efficient than reading the data from Table Store every time.

Example:

```
CREATE TABLE internal_orders AS
SELECT odps_orderkey, odps_orderdate, odps_custkey, odps_totalprice
FROM ots_table_external
WHERE odps_orderkey > 5000 ;
```

> **Note** internal_orders is a common MaxCompute table that has all the features of a MaxCompute internal table. These features include efficient compressed column storage and complete meta. This table is stored in MaxCompute, so it can be accessed faster than an external table in Table Store. This feature is particularly suitable for hot data that is used for multiple computations.

1.17.2.2.3. Export data from MaxCompute to Tablestore

Data interaction between MaxCompute and Tablestore includes importing data from Tablestore to MaxCompute for batch processing and exporting data processing results from MaxCompute to Tablestore. Tablestore features, such as real-time update and single-row overwriting, allow you to quickly upload offline computing results to online applications. The INSERT OVERWRITE statement is used to export data processing results from MaxCompute to Tablestore.

> **Note** MaxCompute does not create external tables in Tablestore. Before you export data to a table in Tablestore, make sure that the table exists in Tablestore. Otherwise, an error is reported.

Assume that a user creates the ots_table_external table in MaxCompute to access data of the ots_tpch_orders table in Tablestore. A data record named internal_orders is stored in MaxCompute. To process the internal_orders data record and write the processing result to Tablestore, execute the INSERT OVERWRITE TABLE statement. Example:

```
INSERT OVERWRITE TABLE ots_table_external
SELECT odps_orderkey, odps_orderdate, odps_custkey, CONCAT(odps_custkey,
'SHIPPED'), CEIL(odps_totalprice)
FROM internal_orders;
```

> **Note**
> - Tablestore is a NoSQL data storage service that stores data in the format of key-value pairs. Data outputs from MaxCompute affect only the rows that contain the primary keys of the Tablestore table. In addition, only the attribute columns specified when you create the table are updated. The columns that are not included in the external table are not modified.
> - When you execute the INSERT OVERWRITE statement on the external table, MaxCompute writes 200 data records in each batch by default. You can adjust a batch size to limit the total batch size to 4 MB.

1.17.2.3. AnalyticDB data source

1.17.2.3.1. Overview

AnalyticDB updates or processes data. If both the data processed by AnalyticDB and the data in MaxCompute are used for computation, the data from AnalyticDB must be synchronized with the data from MaxCompute. To accomplish this, you can create an external table to access the AnalyticDB data.

The following example shows how MaxCompute accesses and processes AnalyticDB data.

1.17.2.3.2. Write data to AnalyticDB

1.17.2.3.2.1. Create an external table

Run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
drop table if exists ads_table_external;
 CREATE EXTERNAL TABLE if not exists ads_table_external
 (
   c_int int ,
  c_tinyint tinyint ,
  c_boolean boolean ,
  c_smallint smallint ,
  c_biging bigint ,
  c_double double ,
  c_float float ,
  --c_time datetime ,
  c_date date ,
  c_timestamp datetime ,
  c_string string
 )
 STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
 location 'jdbc:mysql:host:port/databasename? useSSL=false&user=${user}&password=${password}&
table=${tablename}'
 TBLPROPERTIES(
   'mcfed.mapreduce.jdbc.input.orderby'='c_int'
   )
;
```

> **Note** The preceding command is for reference only.

1.17.2.3.2.2. Write and query data

After an external table is created, you can use it in the same way you would use a MaxCompute table. You can execute the **INSERT OVERWRITE | INTO** and **SELECT** statements to write data and query whether the write operation is successful respectively. For more information about the statements, see *DML statements* in *MaxCompute SQL*.

1.17.2.3.3. Read data from AnalyticDB

Run the following commands to read data from AnalyticDB:

```
set odps.sql.hive.compatible=true;
drop table if exists ads_read_external;
 CREATE EXTERNAL TABLE if not exists ads_read_external
 (
   c_int int ,
  c_tinyint tinyint ,
  c_boolean boolean ,
  c_smallint smallint ,
  c_biging bigint ,
  c_double double ,
  c_float float ,
  --c_time datetime ,
  c_date date ,
  c_timestamp datetime ,
  c_string string
 )
 STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
 location 'jdbc:mysql:host:port/databasename? useSSL=false&user=${user}&password=${password}&
table=${tablename}'
 TBLPROPERTIES(
   'mcfed.mapreduce.jdbc.input.orderby'='c_int'
   )
;
-- Create an external table.
select * from ads_read;
-- Query and read data.
```

Note The preceding commands are for reference only.

1.17.2.4. RDS data source

1.17.2.4.1. Overview

RDS updates or processes data. If both the data processed by RDS and the data in MaxCompute is used in computation, the data in RDS must be synchronized to MaxCompute. In this case, you can access the data in RDS by creating an external table.

The following examples show how MaxCompute accesses and processes RDS data.

Note When you create an external table, the corresponding table may not exist in RDS. However, when you perform the SELECT or INSERT operation on external tables, you must create the corresponding tables in RDS first.

1.17.2.4.2. Write data to RDS

1.17.2.4.2.1. Create an external table

Run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
drop table if exists rds_table_external;
 CREATE EXTERNAL TABLE if not exists rds_table_external
 (
   id bigint,
   name string,
   age tinyint
 )
 STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
 location 'jdbc:mysql:host:port/databasename? useSSL=false&user=${user}&password=${password}&
table=${tablename}'
 TBLPROPERTIES(
   'mcfed.mapreduce.jdbc.input.orderby'='c_int'
   )
;
```

> **Note** The preceding command is for reference only.

1.17.2.4.2.2. Write and query data

After an external table is created, you can use it in the same way you use a MaxCompute table. You can execute the **INSERT OVERWRITE | INTO** and **SELECT** statements respectively to write data and query whether the write operation is successful. For more information about the INSERT OVERWRITE | INTO and SELECT statements, see *DML statements* in *MaxCompute SQL*.

1.17.2.4.3. Read data from RDS

Run the following commands to read data from RDS:

```
set odps.sql.hive.compatible=true;
drop table if exists rds_read_external;
 CREATE EXTERNAL TABLE if not exists rds_read_external
 (
   id int,
   name string,
   age int
 )
 STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
 location 'jdbc:mysql:host:port/databasename? useSSL=false&user=${user}&password=${password}&
table=${tablename}'
 TBLPROPERTIES(
   'mcfed.mapreduce.jdbc.input.orderby'='c_int'
   )
;
-- Create an external table.
select * from rds_read;
-- Query and read data.
```

> **Note** The preceding commands are for reference only.

1.17.2.5. HDFS data source (Alibaba Cloud)

1.17.2.5.1. Overview

Alibaba Cloud Hadoop Distributed File System (HDFS) is a distributed file system designed for Alibaba Cloud computing resources such as ECS and Container Service.

HDFS allows you to manage and access data in the same way as its open-source counterpart. HDFS features such as unlimited capacity, performance scale-out, single namespace, multi-tenancy, high reliability, and high availability, can be used without the need to modify existing big data analysis applications.

MaxCompute can interact with HDFS to jointly compute external tables.

HDFS supports multiple file formats, such as **text file**, **sequence file**, **RC file**, **Parquet**, and **AVRO**. The following example use **text file** to show how MaxCompute accesses and processes HDFS data.

1.17.2.5.2. Data processing for common tables

1.17.2.5.2.1. Write data to HDFS

1.17.2.5.2.2. Read data from HDFS

Run the following command to read data from HDFS after you upload the textfile file to HDFS:

```
set odps.sql.hive.compatible=true;
drop table if exists textfile_external_read;
CREATE external TABLE if not exists textfile_external_read
(
 c_int int ,
 c_tinyint tinyint ,
 c_boolean boolean ,
 c_smallint smallint ,
 c_biging bigint ,
 c_double double ,
 c_float float ,
 c_date date ,
 c_timestamp datetime ,
 c_string string
)
 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
 WITH SERDEPROPERTIES ('field.delim'=',')
 STORED AS textfile
 location "mcfed:dfs://host:port/user/textfile"
-- host must be set as MountPointId.
-- /user/textfile is the file path. Replace it with the actual file path.
 TBLPROPERTIES(
   "mcfed.fs.dfs.impl"="com.alibaba.dfs.DistributedFileSystem"
);
-- Create an external table.
select * from textfile_external_read;
select count(*) from textfile_external_read;
select a.c_int,a.c_boolean,a.c_string,b.value from textfile_external_read a join dfstest b on a.c_int=b.id
;
-- Query and read data.
```

> **Note** The preceding command is for reference only.

1.17.2.5.3. Data processing for partitioned tables

Run the following commands to create an external table and process its data:

```
set odps.sql.hive.compatible=true;
drop table if exists textfile_partition;
CREATE external TABLE if not exists textfile_partition
(
 id string,
 name string
)
 partitioned by (date string)
 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
 WITH SERDEPROPERTIES ('field.delim'=',')
 STORED AS textfile
 location "mcfed:dfs://host:port/user/partition/textfile/"
-- host must be set as MountPointId.
-- user/partition/textfile/ is the file path. Replace it with the actual file path.
 TBLPROPERTIES(
   "mcfed.fs.dfs.impl"="com.alibaba.dfs.DistributedFileSystem"
);
-- Create an external table.
alter table textfile_partition add partition (date='20190218');
alter table textfile_partition add partition (date='20190219');
-- Add partitions.
insert into table textfile_partition partition(date='20190218')select '1','cd' from (select count(*) from tex
tfile_partition)a;
insert into table textfile_partition partition(date='20190219')select '2','gh' from (select count(*) from te
xtfile_partition)a;
-- Write data to HDFS.
select * from textfile_partition;
select count(*) from textfile_partition;
select a.id,a.name,b.value from textfile_partition a join dfstest b on a.id=b.id;
-- Query and read data.
```

Note The preceding commands are for reference only.

1.17.2.6. TDDL data source

1.17.2.6.1. Overview

By encapsulating MySQL, TDDL provides features such as data partitioning, read/write splitting, and failover. In most cases, TDDL can be directly used to access MySQL databases. TDDL also provides Corona connection mode. Corona is a MySQL proxy that follows the standard MySQL protocol and can use JDBC to establish a connection.

MaxCompute can access MySQL databases of TDDL. Built-in StorageHandlers encapsulate native APIs provided by Hadoop such as org, Apache, Hadoop, MapReduce, lib, and db. MySQL JDBC is used for underlying data communication.

The following topics describe how MaxCompute accesses and processes TDDL data.

> **Notice** Among create, read, update, and delete (CRUD) operations, only the following operations are supported:
>
> - MaxCompute reads data from the external table created for a MySQL database.
> - MaxCompute writes data to the external table in the append mode.

1.17.2.6.2. Prerequisites

Because many features are disabled in the MaxCompute 2.0 by default, you must manually configure the following settings:

```
set odps.sql.hive.compatible=true;
-- You must configure this item for all DDL and DML statements to be used on TDDL external tables.
```

```
set odps.sql.udf.java.retain.legacy=false;
-- You must configure this item for all DDL and DML statements to be used on TDDL external tables.
```

```
set odps.sql.jdbc.splits.num=3;
-- Set the number of splits that MaxCompute reads from the MySQL database. Maximum value: 256. De
fault value: 1. You must configure this item for the SELECT operation on TDDL external tables.
```

```
set odps.sql.jdbc.reducer.num=3;
-- Set the number of concurrent instances that MaxCompute writes to the MySQL database. Maximum
value: 256. Default value: 64. If the number of concurrent instances in the generated execution plan is
smaller than this value, no changes are made. You must configure this item for the INSERT operation o
n TDDL external tables.
```

```
set odps.sql.hive.compatible=true;
-- Use an open-source community API to obtain and parse MySQL data types. You must configure this i
tem for all DDL and DML statements to be used on TDDL external tables.
```

```
set odps.sql.type.system.odps2=true;
-- You must configure this item if new data types TINYINT, SMALLINT, INT, FLOAT, VARCHAR, TIMESTAM
P, and BINARY are involved in SQL operations such as CREATE, SELECT, and INSERT.
```

1.17.2.6.3. Create a TDDL external table

1.17.2.6.3.1. Syntax

External tables can act as interfaces between MaxCompute and databases. The method used to process MySQL unstructured data in TDDL is similar to the method to access and process OSS unstructured data. First, you must execute the CREATE EXTERNAL TABLE statement to create an external table. The syntax is as follows:

```
-- Remember to add the corresponding SET statement.
DROP TABLE [IF EXISTS] <external_table_name>;
CREATE EXTERNAL TABLE [IF NOT EXISTS] <external_table_name>
(<column schemas>)
STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
location 'jdbc:mysql://path_format'
TBLPROPERTIES(
  ...
);
```

Description:

- **column schema**: supports the following data types.

Data type mapping

MySQL type	MaxCompute type
TINYINT (unsigned)	TINYINT
SMALLINT (unsigned)	SMALLINT
INT (unsigned)	INT
BIGINT (unsigned)	BIGINT
BOOLEAN	BOOLEAN
FLOAT	FLOAT
DOUBLE	DOUBLE
VARCHAR	VARCHAR
TEXT	STRING
DATE	DATE
DATETIME	DATETIME

MySQL type	MaxCompute type
DECIMAL	DECIMAL (x, y) (The default precision is (10, 0). An error is returned when overflow occurs.)

Notice Because unsigned data types are not supported in MaxCompute, loss of precision may occur if unsigned types are specified.

- `setproject odps.sql.udf.strict.mode=true;` (strict mode, which is the default mode).
  - When reading external tables: MaxCompute can read the data if unsigned data is converted to signed data without loss of precision. A RuntimeException ("value out of range") error is reported if loss of precision occurs during data type conversion.
  - When writing external tables: MaxCompute does not check data types. You can specify the SQL mode to let MySQL produce desired data check actions. For more information about SQL mode settings and data check actions, see Server SQL Modes.
- `setproject odps.sql.udf.strict.mode=false;` (non-strict mode)
  - When reading external tables: MaxCompute can read unsigned data that has been converted to signed data without loss of precision. NULL is obtained if loss of precision occurs during data type conversion.
  - When writing external tables: MaxCompute does not check data types. You can specify the SQL mode to let MySQL produce desired data check actions.

- **STORED BY:** Only built-in StorageHandlers are supported. TDDL table field types must be within the range of supported data types in **column schema**.
- **LOCATION:** Three LOCATION formats are supported.

  1. Access a MySQL database through a JDBC connection string.

  ```
  jdbc:mysql://<user>:<password>@<host>/<databaseName>? useSSL=false&table=<tableName>
  ```

  user and password are the username and password of the JDBC connection string. host is the network address of the MySQL database. databaseName is the name of the MySQL database. tableName is the name of the MySQL table corresponding to the external table.

  2. Access the MySQL database of TDDL through Corona.

  ```
  jdbc:mysql://<user>:<password>@<host>/<databaseName>? useSSL=false&table=<tableName>
  ```

  3. Access the MySQL database of TDDL through an application name.

  ```
  jdbc:mysql://dummy_host? table=<tableName>
  ```

  tableName is the name of the MySQL table corresponding to the external table. You must specify `odps.federation.jdbc.tddl.appname` in the TBLPROPERTIES clause.

> **Notice**
>
> In the first location format, MaxCompute interacts with the database through JDBC. You must enter your username and password as plaintext data, which makes this location format less secure than others. Although usernames and passwords will be hidden when LogView or DESC EXTENDED TABLE is used in MaxCompute, we recommend that you use a separate DDL statement to create an external table before using the external table.
>
> For example, a project member with higher permissions can create an external table in MaxCompute. Other project members can then directly use the external table. This prevents project members with lower permissions from using the plaintext username and password, and prevents the plaintext password from being contained in SQL scripts.

- **TBLPROPERTIES**: includes the following items.
  - **odps.federation.jdbc.condition**: specifies the filter when MaxCompute reads data from a MySQL database. The difference between `odps.federation.jdbc.condition` and `select * from text_test_jdbc_write_external where condition` :

    Suppose the MySQL table contains 100 rows of data and you want to filter the data such that you obtain 10 rows. When you execute `odps.federation.jdbc.condition` , the MySQL table is filtered and MaxCompute only reads 10 rows from the external table. When you execute `select * from text_test_jdbc_write_external where condition` , MaxCompute reads 100 rows from the MySQL table, and then obtains 10 rows.
  - **odps.federation.jdbc.colmapping**: specified column name mapping. Example:

```
-- mysql schema:  mysqlId int
 -- MaxCompute create table
 CREATE EXTERNAL TABLE if not exists table_external
 (
 odpsId1 int,
 odpsId2 int
 )
 STORED BY ...
 location ...
 TBLPROPERTIES('odps.federation.jdbc.colmapping'='odpsId1:mysqlId, odpsId2:mysqlId');
```

  - **odps.federation.jdbc.insert.type**: specifies the insertion type when data is written into the MySQL database. The following data insertion types are supported: simpleInsert, insertOnDuplicateKeyUpdate, and replaceInto. By default, the insertion type is simpleInsert if this parameter is not specified.

    The INSERT statement executed in MaxCompute is parsed into the following SQL statements to update the database:

```
insert into sqlTable xxx values xxx;
 insert into sqlTable xxx values xxx on duplicate key update col1=values(col1), col2=values(col2);
 replace into sqlTable xxx values xxx;
```

- **odps.federation.jdbc.tddl.app.access.key**: the AccessKey ID for the authorized application.
- **odps.federation.jdbc.tddl.app.secret.key**: the AccessKey Secret for the authorized application.
- **odps.federation.jdbc.tddl.appname**: the application name of TDDL. Note that if you specify this value, MaxCompute uses the application name to access the MySQL database in TDDL SDK mode.

1.17.2.6.3.2. Example

The following example shows how to use the application name to access the MySQL database of TDDL. In this example, the application name is ODPS_TDDL_TEST_APP and the table name is odps_federation_localrun_write.

Example:

```
-- Remember to add the corresponding SET statement.
drop table if exists text_test_jdbc_external;
CREATE EXTERNAL TABLE if not exists text_test_jdbc_external
(
 colmapping tinyint, --c_tinyint tinyint,
 c_smallint smallint,
 c_int int,
 c_bigint bigint,
 c_utinyint tinyint,
 c_usmallint smallint,
 c_uint int,
 c_ubigint bigint,
 c_boolean tinyint,
 --c_float float,  -- in tddl, not recommend float and double type as it may lost precision
 --c_double double,
 c_string string,
 c_datetime datetime,
 c_decimal decimal
)
STORED BY 'com.aliyun.odps.jdbc.JdbcStorageHandler'
location 'jdbc:mysql://dummy_host? table=odps_federation_localrun_write'
TBLPROPERTIES(
'odps.federation.jdbc.insert.type'='simpleInsert',
'odps.federation.jdbc.condition'='c_boolean = 1 and c_int is not null and c_utinyint=127',
'odps.federation.jdbc.colmapping'='colmapping:c_tinyint',
'odps.federation.jdbc.tddl.appname'='ODPS_TDDL_TEST_APP',
'odps.federation.jdbc.tddl.app.access.key'='your tddl app access key',
'odps.federation.jdbc.tddl.app.secret.key'='your tddl app secret key');
```

1.17.2.6.4. Read data from an external table

For complex operations such as GROUP JOIN, we recommend that you import data from external table to MaxCompute tables before performing operations. This improves the efficiency of data computation. The following example shows how to import data from an associated MySQL external table to MaxCompute.

Create a MaxCompute table

Example:

```
CREATE TABLE if not exists text_test_jdbc_max_compute
(
 c_tinyint tinyint,
 c_smallint smallint,
 c_int int,
 c_bigint bigint,
 c_utinyint tinyint,
 c_usmallint smallint,
 c_uint int,
 c_ubigint bigint,
 c_boolean tinyint,
 --c_float float,
 --c_double double,
 c_string string,
 c_datetime datetime,
 c_decimal decimal
);
```

Import data to a MaxCompute table

Example:

```
-- Remember to add the SET statement.
insert OVERWRITE TABLE text_test_jdbc_odps select * from text_test_jdbc_read_external;
```

Relationship between creating an external table and importing data to a MaxCompute table

When you create an external table, only a data channel is established between MaxCompute and MySQL. MaxCompute does not store any MySQL data. If external table data is lost from the MySQL database, it will not be available in MaxCompute.

When data is imported to a MaxCompute table, the data is actually stored in the MySQL database. If imported data is lost from the MySQL database, it can be retrieved from the MaxCompute table.

1.17.2.6.5. Write data to an external table in the append mode

The column names and data types of the external table must be consistent with those of the database to ensure that the correct data is written to the eternal table. For more information about data check actions when loss of precision occurs during data type conversion, see the column schema parameter of Syntax.

An example of the command used is as follows:

```
-- Remember to add the SET statement.
insert INTO TABLE  text_test_jdbc_external select * from text_test_jdbc_max_compute;
```

> **Note** For MySQL external tables, **insert INTO** mysql-external-table uses the same syntax as **insert OVERWRITE** mysql-external-table. No matter which statement is executed, data is appended to the table and you can use `ODPS. federation. jdbc. insert. type` to specify the data insertion type. For more information, see the **TBLPROPERTIES** parameter in Syntax. However, the preceding syntax notes are not applicable to MaxCompute tables.

1.17.3. External data sources

1.17.3.1. HDFS data source (open-source)

1.17.3.1.1. Overview

HDFS is the most widely used storage service in the open-source community. Most customers use HDFS at the underlying layer of their self-developed big data systems.

MaxCompute uses external tables to access HDFS data to facilitate data migration, interact with self-developed customer systems, and reduce the efforts and costs of customers.

HDFS supports multiple file formats, such as **text file**, **sequence file**, **RC file**, **Parquet**, and **AVRO**. The following example use **text file** to show how MaxCompute accesses and processes HDFS data.

1.17.3.1.2. Write data to HDFS

1.17.3.1.2.1. Create an external table

Run the following command to create an external table after the testfile script has been compiled:

```
set odps.sql.hive.compatible=true;
drop table if exists textfiletest;
 CREATE EXTERNAL TABLE if not exists textfiletest
 (
  c_int int ,
  c_tinyint tinyint ,
  c_boolean boolean ,
  c_smallint smallint ,
  c_biging bigint ,
  c_double double ,
  c_float float ,
  --c_time datetime ,
  c_date date ,
  c_timestamp datetime ,
  c_string string
 )
 STORED as TEXTFILE
 location 'hdfs://host:port/user/wbyy/';
-- File path /user/wbyy/ is for reference only. Replace it with the path to actually be accessed.
```

> **Note** The preceding command is for reference only.

1.17.3.1.2.2. Write and query data

After an external table is created, you can use it in the same way you would use a MaxCompute table. You can execute the **INSERT OVERWRITE | INTO** and **SELECT** statements to write data and check whether the write operation is successful respectively. For more information about the statements, see *DML statements* in *MaxCompute SQL*.

1.17.3.1.3. Read data from HDFS

Run the following command to read data from HDFS after you compile the testfile script:

```
set odps.sql.hive.compatible=true;
drop table if exists testfile_read;
 CREATE EXTERNAL TABLE if not exists testfile_read
 (
  c_int int ,
  c_tinyint tinyint ,
  c_boolean boolean ,
  c_smallint smallint ,
  c_biging bigint ,
  c_double double ,
  c_float float ,
  --c_time datetime ,
  c_date date ,
  c_timestamp datetime ,
  c_string string
 )
 STORED as TEXTFILE
 location 'hdfs://host:port/user/wbyy/';
-- File path /user/wbyy/ is for reference only. Replace it with the path to actually be accessed.
-- Create an external table.
select * from testfile_read;
-- Query and read data.
```

> **Note** The preceding command is for reference only.

1.17.3.2. MongoDB data source

1.17.3.2.1. Overview

ApsaraDB for MongoDB is a stable, reliable, and auto-scaling database service that is fully compatible with MongoDB protocols. MongoDB offers a full range of database solutions, such as disaster recovery, backup, restoration, monitoring, and alerting.

MaxCompute can interact with MongoDB for joint computation after you create external tables.

The following examples show how MaxCompute accesses and processes MongoDB data.

1.17.3.2.2. Prerequisites

You must first deploy MongoDB before creating an external table and processing MongoDB data.

1. Run the following command to enable the MongoDB service:

   ```
   bin/mongod --dbpath=./db
   ```

2. Run the following command to start the MongoDB client:

```
bin/mongo --host=${host}
```

3. Run the following command to create a database:

```
use mongodb
```

4. Run the following command to create a username and password:

```
db.createUser({user: '${user}', pwd: '${password}', roles: [{role:'readWrite',db:'mongodb'}]})
```

5. Run the following command to check whether the operation is successful. A response of 1 indicates a successful operation.

```
db.auth('${user}', '${password}')
```

1.17.3.2.3. Write data to MongoDB

1.17.3.2.3.1. Create an external table

Run the following command to create a collection in MongoDB:

```
db.createCollection("${tablename}", { capped : true, autoIndexId : true, size : 6142800, max : 10000 } )
-- The values of the size and max parameters are for reference only. Replace them with the values to actually be used.
```

After the collection has been created, run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
drop table if exists mongo_table_external;
CREATE external TABLE if not exists mongo_table_external
   (
    id string,
    name string
   )
   STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
   location "mcfed:mongodb://${user}:${password}@host:port/mongodb.${tablename}"
   TBLPROPERTIES(
      "mcfed.mongo.input.split_size"="2",
-- input.split_size value is for reference only. Replace them with the values to actually be used.
      "mcfed.location"="mongodb://${user}:${password}@host:port/mongodb.${tablename}",
      "mcfed.mongo.input.uri"="mongodb://${user}:${password}@host:port/mongodb.${tablename}",
      "mcfed.mongo.output.uri"="mongodb://${user}:${password}@host:port/mongodb.${tablename}"
);
```

> **Note** The preceding commands are for reference only.

1.17.3.2.3.2. Write and query data

After an external table is created, you can use it in the same way that you would use a MaxCompute table. You can execute the **INSERT OVERWRITE | INTO** and **SELECT** statements to write data and check whether the write operation is successful respectively. For more information about the statements, see *DML statements* in *MaxCompute SQL*.

1.17.3.2.4. Read data from MongoDB

Run the following command to read data from MongoDB after a row of data has been inserted into a created collection:

```
set odps.sql.hive.compatible=true;
drop table if exists mongo_read_external;
CREATE external TABLE if not mongo_read_external
  (
   id string,
   name string
  )
  STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
  location "mcfed:mongodb://${user}:${password}@host:port/mongodb.${tablename}"
  TBLPROPERTIES(
     "mcfed.mongo.input.split_size"="2",
-- The value of the input.split_size parameter is for reference only. Replace it with the value to actuall
y be used.
     "mcfed.location"="mongodb://${user}:${password}@host:port/mongodb.${tablename}",
     "mcfed.mongo.input.uri"="mongodb://${user}:${password}@host:port/mongodb.${tablename}"
);
-- Create an external table.
select * from mongo_external;
-- Query and read data.
```

> **Note** The preceding command is for reference only.

1.17.3.3. HBase data source

1.17.3.3.1. Overview

ApsaraDB for HBase is a distributed database based on Hadoop. It can store PBs of data and be used in scenarios requiring high-throughput random read/writes.

MaxCompute can interact with HBase for joint computation after you create external tables.

The following examples show how MaxCompute accesses and processes HBase data.

1.17.3.3.2. Write data to HBase

1.17.3.3.2.1. Create an external table

Run the following command to create an external table:

```
set odps.sql.hive.compatible=true;
drop table if exists hbase_table_external;
CREATE EXTERNAL TABLE if not exists hbase_table_external
(
id string,
cfa string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('mcfed.hbase.table.name'='${table.name}','mcfed.hbase.columns.mapping':key,cf:a')
-- cf is the column family in the HBase table.
location 'hbase://host:port'
TBLPROPERTIES('hbase.table.name'='${table.name}','hbase.columns.mapping':key,cf:a', 'mcfed.zookeeper.session.timeout'='30', 'mcfed.hbase.client.retries.number'='1', "mcfed.hbase.zookeeper.quorum"="${host}", "mcfed.hbase.zookeeper.property.clientPort"="${port}");
-- The values of the zookeeper.session.timeout and hbase.client.retries.number parameters are for reference only. Replace them with the values to actually be used.
```

> **Note** The preceding command is for reference only.

1.17.3.3.2.2. Write and query data

After an external table is created, you can use it in the same way that you would use a MaxCompute table. You can execute the **INSERT OVERWRITE | INTO** and **SELECT** statements to write data and check whether the write operation is successful respectively. For more information about the statements, see *DML statements* in *MaxCompute SQL*.

1.17.3.3.3. Read data from HBase

Run the following commands to read data from HBase after you have created a table in the HBase client and inserted data into it:

```
set odps.sql.hive.compatible=true;
drop table if exists hbase_read_external;
CREATE EXTERNAL TABLE if not exists hbase_read_external
(
id string,
name string,
a string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ('mcfed.hbase.table.name'='${table.name}','mcfed.hbase.columns.mapping'=':key,f1:name,f1:a')
-- f1 is the column family in the HBase table.
location 'hbase://host:port'
TBLPROPERTIES('hbase.table.name'='${table.name}','hbase.columns.mapping'=':key,f1:name,f1:a', 'mcfed.zookeeper.session.timeout'='30', 'mcfed.hbase.client.retries.number'='1', "mcfed.hbase.zookeeper.quorum"="${host}", "mcfed.hbase.zookeeper.property.clientPort"="${port}");
-- The values of the zookeeper.session.timeout and hbase.client.retries.number parameters are for reference only. Replace them with the values to actually be used.
-- Create an external table.
select * from hbase_read_external;
select count(*) from hbase_read_external;
select a.id,a.name from hbase_read_external a join hbase_test b on a.id=b.id;
-- Query and read data.
```

> **Note** The preceding commands are for reference only.

1.18. Unstructured data access and processing (inside MaxCompute)

1.18.1. Overview

MaxCompute has the following problems when processing unstructured data: MaxCompute stores data as volumes and must export generated unstructured data to an external system for processing.

To alleviate these problems, MaxCompute uses external tables to enable connections between MaxCompute and various data types. MaxCompute uses external tables to read and write data volumes as well as process unstructured data from external sources such as OSS.

The following topics describe how MaxCompute accesses and processes volume unstructured data through external tables.

1.18.2. Create a volume external table

1.18.2.1. Syntax

You must execute the CREATE EXTERNAL TABLE statement to create an external table.

```
DROP TABLE [IF EXISTS] <external_table_name>;
CREATE EXTERNAL TABLE [IF NOT EXISTS] <external_table_name>
(<column schemas>)
[PARTITIONED BY (partition column schemas)]
STORED BY '<StorageHandler>'
[WITH SERDEPROPERTIES (
                'name'='value'
                )]
LOCATION 'volume://...'
[USING '<Resourcename>']
;
```

Description:

- **STORED BY**: Two built-in StorageHandlers `com.aliyun.odps.CsvStorageHandler` and `com.aliyun.odps.TsvStorageHandler` are supported. They can be used to read and write CSV files where the column delimiter is a comma and the row delimiter is `\n` or TSV files where the column delimiter is `\t` and the row delimiter is `\n`). If the built-in StorageHandlers cannot be used for some reason, you can build a custom StorageHandler.
- **WITH SERDEPROPERTIES**: specifies table attributes such as delimiters for a custom StorageHandler.
- **LOCATION**: the location format of the table.

  Format:

  ```
  volume://[project_name]/volume_name/partition_value
  ```

  Example:

  ```
  volume://test_project/volume_data/20190102
  ```

  project_name is optional. If **project_name** is not specified, the current project is used to obtain volume data after the DML SQL statement is executed. In the preceding example, if the current project is myproject, you can use the following location format:

  ```
  volume:///volume_data/20190102
  ```

> **Notice**
>
> - The location of a non-partitioned table must point to a volume partition, instead of the volume itself.
> - The location of a partitioned table must point to the volume itself.
> - The volume path cannot contain an equal sign (=) and does not support the default standard partition path *ds=2017071* that is used when a partition is created. The partition path must be customized. The custom partition path can be any path supported by the volume. For example, if the partition path is 20190102, the path combined with volume path can be *volume://test_project/volume_data/20190102*.

- **USING**: specifies the StorageHandler resource. To use a custom StorageHandler, you must first export the custom StorageHandler as a JAR package and then add it to MaxCompute as a JAR resource.

1.18.2.2. Use the built-in StorageHandler to create an external table

You can use the built-in StorageHandler to create a partitioned or non-partitioned table.

Create a non-partitioned table

Example:

```
DROP TABLE IF EXISTS volume_ext;
CREATE EXTERNAL TABLE volume_ext
(
key string,
value string
)
STORED BY 'com.aliyun.odps.CsvStorageHandler'--The built-in StorageHandler.
LOCATION 'volume://test_project/volume_data/20190102'
;
```

Create a partitioned table

Example:

```
DROP TABLE IF EXISTS volume_ext_pt;
CREATE EXTERNAL TABLE volume_ext_pt
(
key string,
value string
)
PARTITIONED BY (ds string)
STORED BY 'com.aliyun.odps.CsvStorageHandler'--The built-in StorageHandler.
LOCATION 'volume://test_project/volume_data'
;
ALTER TABLE volume_ext_pt DROP IF EXISTS PARTITION (ds="20190102");
ALTER TABLE volume_ext_pt ADD PARTITION (ds="20190102") LOCATION "volume://test_project/volume
_data/20190102";
```

1.18.2.3. Use a custom StorageHandler to create a table

When the built-in StorageHandlers are unable to meet the requirements of your business, you can customize a StorageHandler through Java and specify some attributes of the Volume external table which you want to process data through.

The following example shows how to use a custom StorageHandler to create an external table.

Assume that the data type is TEXT and the column delimiter is "|". You can perform the following steps to create an external table:

1. Use MaxCompute Studio or MaxCompute Eclipse development plug-in to customize various Java classes.
2. Export the JAR package. In this example, the package name is **odps-volume-example.jar**.
3. Run the following command to add the JAR package to MaxCompute as a resource:

   ```
   add jar odps-volume-example.jar -f;
   ```

4. Run the following commands to create an external table:

```
DROP TABLE IF EXISTS volume_ext;
CREATE EXTERNAL TABLE volume_ext
(
key string,
value string
)
STORED BY 'com.aliyun.odps.udf.example.text.TextStorageHandler'
WITH SERDEPROPERTIES (
                'delimiter'='|'
                )
LOCATION 'volume://myproject/volume_data/20190102'
USING 'odps-volume-example.jar'
  ;
```

Note After the external table is created, you can operate the volume data through the external table.

1.18.3. Access a volume external table

Volume external tables can be accessed in the same way that you would access a MaxCompute table. Example:

```
select key,value from volume_ext_pt where ds="20190102";
```

1.19. Multi-region cluster deployment on MaxCompute

1.19.1. Overview

MaxCompute supports multi-region cluster deployment.

The control clusters is deployed in a centralize manner and configures resources as well as manages compute tasks. The Compute cluster is deployed independently for each region and handles project creation and compute task delegation.

1.19.2. Characteristics of multi-region deployment

This topic describes the characteristics of multi-region deployment.

The multi-region deployment on MaxCompute has the following features:

- One MaxCompute service instance can manage multiple clusters that are located in different cities.
- Cross-cluster data interaction is implemented within the MaxCompute service, and cross-cluster data replication and synchronization are managed based on the actual configuration.

- Metadata is stored in a centralized manner. Therefore, the infrastructure requirements, such as network connections between different data centers, are relatively high.
- A unified account system is required.
- Big data application development systems, such as DataWorks, are used in all regions and clusters.
- You must change the mode of the current MaxCompute cluster to cross-cluster for multi-region deployment.

  > **Note** The conditions and limits on changing the cluster mode are as follows:
  >
  > - The network bandwidth must be sufficient for multi-region data synchronization and link redundancy.
  > - The central-region control cluster has a relatively high latency for basic services such as DNS and TableStore. We recommend that you deploy basic services in the same data center to keep the network latency within 5 ms.
  > - The network latency between the central-region control cluster and computing clusters in other regions must must be kept within 20 ms.
  > - Clocks must be synchronized between clusters in different cities and between machines in the same cluster.
  > - The network bandwidth must be sufficient for data replication among clusters.
  > - DNS is required.
  > - In order for cross-cluster servers to communicate on the same network, the clusters must have similar network bandwidth conditions, such as GE or 10GE.

- The operations, maintenance, deployment, and upgrade for multi-region deployment are different from those for a single cluster deployment. Multi-region deployment requires higher onsite operations and maintenance capabilities.

1.19.3. Instructions on multi-region deployment

This topic describes multi-region deployment.

The instructions to implement multi-region deployment are as follows:

- Computing tasks on MaxCompute are implemented in a cluster based on data distribution within the service. Cross-cluster replication and direct reads are performed based on the cluster configuration and distribution of data.
- Data replication and synchronization can be performed between different clusters by table or partition based on the actual cluster configuration.
- You must plan the relationship between the MaxCompute project and the cluster at the business layer. You can select one or multiple clusters for a single project.

  > **Note** If you select multiple clusters for a project, data replication is carried out among these clusters.

- If the project contains business that cannot implement cross-cluster computing, you can distribute data to different clusters and divide the computing tasks based on their distribution.

> Note Cross-cluster data replication and management requires management policies.

- The bandwidth between clusters in different remote data centers must be sufficient for data replication or read-only operations.
- Replication and read-only operations across clusters in MaxCompute involving large amounts of data will consume all of the bandwidth between the two clusters, as the two data centers share bandwidth.
- A cluster failure in the primary data center will affect the entire service because the primary data center stores information such as metadata and accounts.
- When the secondary data center fails, the project where data is stored on the cluster and the O&M of services are affected.

> Note If project data is independently stored in the primary data center, the failure of the secondary data center will not affect computing tasks of the project.

- In general scenarios, data is partitioned and replicated based on business characteristics, and computing tasks are performed based on data types in different clusters. The combined results of different MaxCompute tasks are applied.

1.19.4. Multi-region deployment examples

1.19.4.1. Synchronize table data among multiple clusters

This topic provides an example on how to synchronize table data among multiple clusters in multi-region deployment scenarios.

Prerequisites

- When you create a project in the AdminConsole, you have selected two clusters and set one as the default cluster. The following figure shows an example.

> Note
> - In this example, the project is multiregion, and AT-MDU-TEST and AT-5KN are selected for it, with AT-MDU-TEST as the default cluster.
> - The URL of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**, which is port 9090 of MaxCompute AG.
> - The operation entry to create the project is **MaxCompute Configuration > Project Management > Create Project**.

- You have configured cross-cluster replication for the project. The following example demonstrates how to configure it.
  i. Choose **MaxCompute Configuration > Global Cross-cluster Replication Configuration** to go to the **Global Cross-cluster Replication Configuration** page.
  ii. In the Add Item part of the **InfoBetweenClusters** section, set key to **AT-5KN#AT-MDU-TEST**, set both availableBandwith and totalBandwith to **2000**, and click **add**.
  iii. Choose **MaxCompute Configuration > Project Management**. Find multiregion and click **Cross-cluster Replication Configuration**.

iv. In the dialog box that appears, configure the parameters, as shown in the following figure.

v. Click **Save** and then click **Start Replication**.

Procedure

Example:

1. In the AG of the default cluster AT-MDU-TEST, construct an upload tunnel data file in the directory of the same level as console, such as `echo "testtest" > uploaddata`.
2. Go to the console command line and run the following commands to create a table and insert data into the table:

```
use multiregion;
create table t1 (s string);
tunnel upload uploaddata t1;
```

3. Run the following command to check whether the data is inserted into the table:

```
select * from t1;
```

4. In the AG of AT-MDU-TEST, view the Apsara Distributed File System directory for this table. Run the following command to check whether data exists in the directory:

```
pu ls product/aliyun/odps/multiregion/data/t1
```

5. In the AG of AT-5KN, view the Apsara Distributed File System directory for this table. Run the following command to check whether data exists in the directory:

```
pu ls /apsara/odps/mdutesting/multiregion/data/t1
```

6. If data is displayed after you perform both steps 4 and 5, the cross-cluster data synchronization operation is successful.

1.19.4.2. Query the status of data synchronization between primary and secondary clusters

This topic provides an example on how to query the status of data synchronization between the primary and secondary clusters in multi-region deployment scenarios.

Prerequisites

- When you create a project in the AdminConsole, you have selected two clusters and set one as the default cluster. The following figure shows an example.

> **Note**
>
> - In this example, the project is multiregion, and AT-MDU-TEST and AT-5KN are selected for it, with AT-MDU-TEST as the default cluster.
> - The URL of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**, which is port 9090 of MaxCompute AG.
> - The operation entry to create the project is **MaxCompute Configuration > Project Management > Create Project**.

- You have configured cross-cluster replication for the project. The following example demonstrates how to configure it.
  i. Choose **MaxCompute Configuration > Global Cross-cluster Replication Configuration** to go to the **Global Cross-cluster Replication Configuration** page.
  ii. In the Add Item part of the **InfoBetweenClusters** section, set key to **AT-5KN#AT-MDU-TEST**, set both availableBandwith and totalBandwith to **2000**, and click **add**.
  iii. Choose **MaxCompute Configuration > Project Management**. Find multiregion and click **Cross-cluster Replication Configuration**.
  iv. In the dialog box that appears, configure the parameters, as shown in the following figure.
  v. Click **Save** and then click **Start Replication**.

Procedure

Example:

1. Compile the bodychecksync configuration file, which is used to send admintask.

```
<Instance>
 <Job>
  <Comment>
  </Comment>
  <Priority>1</Priority>
  <Tasks>
   <Admin>
   <Name>task_1</Name>
   <Comment>test</Comment>
   <Config>
    <Property>
     <Name>PROJECT</Name>
     <Value>multiregion</Value>
    </Property>
    <Property>
     <Name>CLUSTER</Name>
     <Value>AT-MDU-TEST</Value>
    </Property>
   </Config>
   <Command>GET_UNREPLICATED_OBJECTS</Command>
   </Admin>
  </Tasks>
  <DAG>
   <Comment/>
   <RunMode>Sequence</RunMode>
  </DAG>
 </Job>
</Instance>
```

2. Compile the header.

```
content-type: application/xml
```

3. Run the following command to send admintask:

```
CLTrelease/bin/odpscmd -e "http post  /projects/admin_task_project/instances -header=header -
content=bodychecksync;"
```

4. Run the following command to check the instance execution results:

```
CLTrelease/bin/odpscmd  --project=admin_task_project -e "wait 20180711050550317gnege3ms2;"
```

5. Access the LogView URL generated in step 4 in a browser.
6. On the page that appears, click **Detail**. The status of data synchronization between the

primary and secondary clusters is displayed.

1.19.4.3. Cross-region direct read

This topic provides examples of cross-region direct read operations in multi-region deployment scenarios.

Cross-region direct read can be implemented with or without cross-cluster access.

> **Note** When cross-cluster access is not configured and the data volume is large, cross-region direct read takes a long period of time.

Direct read when cross-cluster access is not configured

Example:

1. In the AdminConsole, create two projects: testmdu and testcross5kn.

   > **Note**
   > - In this example, AT-MDU-TEST and AT-5KN are selected for testmdu, with AT-MDU-TEST as the default cluster. AT-5KN is selected for testcross5kn and serves as the default cluster.
   > - The URL of Apsara Stack MaxCompute AdminConsole is **http://{odps_ag}:9090**, which is port 9090 of MaxCompute AG.
   > - The operation entry to create the projects is **MaxCompute Configuration > Project Management > Create Project**.

2. Create tables for testmdu and testcross5kn.

   ```
   create table testrep(s string );
   create table tablecross5kn (s string );
   ```

3. Construct some data for tables testrep and tablecross5kn.
4. Run the following commands to read the table data in testmdu directly from testcross5kn:

   ```
   use testcross5kn;
   select * from testmdu.testrep;
   ```

   > **Note** If the specified data is returned, the cross-region direct read operation is successful.

Direct read when cross-cluster access is configured

Example:

1. In the AdminConsole, create two projects: testmdu and testcross5kn.

> **Note**
>
> - In this example, AT-MDU-TEST and AT-5KN are selected for testmdu, with AT-MDU-TEST as the default cluster. AT-5KN is selected for testcross5kn and serves as the default cluster.
> - The URL of Apsara Stack MaxCompute AdminConsole is http://{odps_ag}:9090, which is port 9090 of MaxCompute AG.
> - The operation entry to create the projects is **MaxCompute Configuration** > **Project Management** > **Create Project**.

2. Create tables for testmdu and testcross5kn.

```
create table testrep(s string );
create table tablecross5kn (s string );
```

3. Construct some data for tables testrep and tablecross5kn.
4. Configure cross-cluster replication for testmdu.

> **Note** Choose **MaxCompute Configuration** > **Project Management**. Find testmdu and click **Cross-cluster Replication Configuration**.

5. Run the following commands to read the table data in testmdu directly from testcross5kn:

```
use testcross5kn;
select * from testmdu.testrep;
```

> **Note** If the specified data is returned, the cross-region direct read operation is successful.

1.19.4.4. Cross-region JOIN

This topic provides an example of cross-region JOIN in multi-region deployment scenarios.

Example:

1. In the AdminConsole, create two projects: crosssregion and crossregion02.

> **Note**
>
> - In this example, AT-MDU-TEST and AT-70N are selected for crosssregion, with AT-MDU-TEST as the default cluster. AT-70N is selected for crossregion02 and serves as the default cluster.
> - The URL of Apsara Stack MaxCompute AdminConsole is http://{odps_ag}:9090, which is port 9090 of MaxCompute AG.
> - The operation entry to create the projects is **MaxCompute Configuration** > **Project Management** > **Create Project**.

2. Configure cross-cluster replication for crosssregion.

> Note Choose MaxCompute Configuration > Project Management. Find crosssregion and click Cross-cluster Replication Configuration.

3. Create a table named business for crosssregion.

```
create table business(bid bigint,name string,phone string,address string,region string);
```

4. Construct some data for the table.
5. Run the following command to import data into the table:

```
tunnel upload business business;
```

6. Create a table named product for crossregion02.

```
create table if not exists product(pid bigint,name string,type string,color string,bid bigint);
```

7. Construct some data for the table.
8. Run the following command to import data into the table:

```
tunnel upload product product;
```

9. Run the following command in the console to obtain specific data from the preceding two tables:

```
select pro.pid,pro.name,pro.type,pro.color,bus.name,bus.phone from crossregion02.product pro joi
n crosssregion.business bus on pro.bid=bus.bid where bus.region='Shanghai';
```

> Note If the specified data is returned, the cross-region JOIN operation is successful.

1.20. Security solution

1.20.1. Target users

This User Guide is intended for all owners and administrators of MaxCompute projects, and users interested in the MaxCompute multi-tenant data security system. The MaxCompute multi-tenant data security system includes:

- User authentication
- User and authorization management of projects
- Cross-project resource sharing
- Project protection

1.20.2. Quick start

Add a user and grant permissions to the user

Scenario: Jack is the administrator of the prj1 project. A new team member Alice, who already has an Alibaba Cloud account (alice@aliyun.com), applies to join the project. Alice applies for the following permissions: viewing table lists, submitting jobs, and creating tables.

The admin performs the following operations to add Alice to the project:

```
use prj1
add user aliyun$alice@aliyun.com;
-- Add a user.
grant List, CreateTable, CreateInstance on project prj1 to user aliyun$alice@aliyun.com
-- Grant permissions to the user.
```

Add a user and grant permissions to the user using an ACL

Scenario: Jack is the administrator of prj1. Three new members Alice, Bob, and Charlie join in as data reviewers. They require the following permissions: viewing table lists, submitting jobs, and reading the table userprofile.

The project administrator can use object-based ACL authorization in this scenario.

The operations are as follows:

```
use prj1
add user aliyun$alice@aliyun.com
-- Add a user.
add user aliyun$bob@aliyun.com
add user aliyun$charlie@aliyun.com
create role tableviewer
-- Create a role.
grant List, CreateInstance on project prj1 to role tableviewer; --Grant permissions to the role
-- Grant permissions to the role.
grant Describe, Select on table userprofile to role tableviewer
grant tableviewer to aliyun$alice@aliyun.com
-- Grant the tableviewer role to a user.
grant tableviewer to aliyun$bob@aliyun.com
grant tableviewer to aliyun$charlie@aliyun.com
```

Package and share resources

Scenario: Jack is the administrator of prj1. John is the administrator of prj2. Due to business requirements, Jack wants to share some resources of prj1 (such as datamining.jar and sampletable) to John's prj2. A user in prj2 (Bob) requires access to these resources. The prj2 administrator can configure an ACL or policy to automatically authorize prj2 users to access these resources, without the intervention of Jack.

The operations are as follows:

1. Prj1 administrator Jack creates a resource package in prj1.

```
use prj1
create package datamining
-- Create a package.
add resource datamining.jar to package datamining
-- Add resources to the package.
add table sampletable to package datamining
-- Add the table to the package.
allow project prj2 to install package datamining
-- Share the package to prj2.
```

2. Prj2 administrator Bob installs the package in prj2.

```
use prj2
install package prj1.datamining
-- Install the package.
describe package prj1.datamining
-- View the resource list of the package.
```

3. Configure automatic authorization for Bob on the package.

```
use prj2
grant Read on package prj1.datamining to user aliyun$bob@aliyun.com
-- Use an ACL to allow Bob to use the package.
```

> **Note** For more information about cross-project resource sharing, see Cross-project resource sharing.

Configure project protection

Scenario: Jack is the administrator of project prj1. This project contains sensitive data such as user IDs and shopping records. The project also stores many data mining algorithms to which the organization holds intellectual property rights. Jack wants to protect the sensitive data and algorithms in the project. He wants the data to be accessible only to users in the project. The data must not be able to flow out of the project.

The operations are as follows:

```
use prj1
set ProjectProtection=true
-- Enable project protection.
```

When project protection is enabled, data in the project can flow only within the project. Data cannot flow out. In some cases, for example, a user (Alice) requires to export data tables for business purposes. This operation is approved by the project administrator. MaxCompute provides two methods to export data from a protected project.

Method 1: Create an exception policy. For more information, see Data export methods when project protection is enabled.

1. Create a policy file. Create a policy file named */tmp/exception_policy.txt*. It only allows Alice to export t1 from prj1 using a SQL task. The policy is defined as follows:

```
{
"Version": "1",
"Statement": [{
"Effect":"Allow",
"Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:Describe","odps:Select"],
"Resource":"acs:odps:*:projects/prj1/tables/t1",
"Condition":{
"StringEquals": { "odps:TaskType":"SQL"
}
}
}]
```

2. Configure the exception policy.

```
use prj1
-- Enable project protection and configure an exception policy.
set ProjectProtection=true with exception /tmp/exception_policy.txt
```

> **Note** When you configure the exception policy, ensure that the principal cannot update the data resources or recreate an object with the same name (using DROP TABLE and CREATE TABLE). This prevents data leakage due to time-of-check to time-of-use (TOC2TOU).

Method 2: Configure trusted projects. Configure prj2 as a trusted project of prj1 to enable data flow from prj1 to prj2. For more information, see Data export methods when project protection is enabled.

```
use prj1
        add trustedproject prj2
```

> **Note** In MaxCompute, package-based resource sharing and project protection are mutually independent mechanisms that take effect at the same time, but their functions are mutually restrictive.

In MaxCompute, resource sharing has a higher priority than project protection. This means, if an object in a protected project is shared with other projects through the package mechanism, cross-project access to this object is not subject to the project protection rules.

1.20.3. User authentication

The main purpose of user authentication is to verify the identity of a request sender. Authentication typically includes:

- Verifying the true identity of a message sender
- Checking whether the message was tampered with before it is received.

1.20.4. Project user and authorization management

1.20.4.1. Overview

Projects are the foundation of the MaxCompute multi-tenant system and the basic units of data management and computing. When you create a project, you are automatically the project owner. All objects in the project, such as tables, instances, resources, and UDFs, belong to you. Objects in the project can only be accessed by the owner and users that are authorized by the owner.

This topic describes users, roles, and authorization management of projects. For example, Alice is the owner of test_project, and another user from Alice's project team requests to access the resources in test_project. Alice can use the methods described in this topic to perform user and authorization management. If a user that wants access to Alice's project is not from her project team, Alice can implement cross-project sharing. For more information, see Cross-project resource sharing.

1.20.4.2. User management

Add a user

If Alice (the project owner) decides to authorize another user, she must add the user to this project. Only users in a project can be authorized.

Run the following command to add a user:

```
add user <full_username>
-- Add a user to a project.
```

Remove a user

When a user leaves the project team, Alice needs to remove the user from the project. After a user is removed from the project, the user no longer has any access permissions on project resources.

Run the following command to remove a user from a project:

```
remove user <full_username>
-- Remove a user from a project.
```

Note

- After a user is removed, the user no longer has any access permissions on project resources.
- Before you remove a user who has been assigned a role, you must first revoke the role. For information about roles, see Role management.
- After a user is removed, the ACL authorization related to the user is retained. However, the policy authorization at the role level is revoked, and the policy authorization at the project level is retained. If the user is added to the project again, the previous ACL authorization of the user is re-activated.
- MaxCompute does not support complete removal of a user and the relevant authorization data.

1.20.4.3. Role management

A role is a collection of access permissions. A role can be used to assign the same permissions to a group of users. Role-based authorization can greatly simplify the authorization process and reduce authorization management costs. When granting permissions to users, you should consider using role-based authorization.

An admin role is automatically created when a project is created. This role is granted permissions to access all objects of the project, manage users and roles, and authorize users and roles. Compared with the project owner, the admin role cannot assign another user with the admin role, configure security rules for a project, or change the authentication model of the project. Permissions of the admin role cannot be modified.

The role management commands are as follows:

```
create role <rolename>
-- Create a role.
drop role <rolename>
-- Delete a role.
grant <rolename> to <username>
-- Assign a role to a user.
revoke <rolename> from <username>
-- Revoke the role of a user.
```

Note When you delete a role, MaxCompute checks whether there are users assigned with this role. If the role is assigned to users, the role fails to be deleted. To delete the role, you must revoke this role from all users.

1.20.4.4. ACL authorization actions

Authorization usually involves three elements: subject, object, and action. In MaxCompute, a subject is the user, there are various types of objects in a project, and actions are performed on objects. Different types of objects support different actions.

MaxCompute projects support the following object types and actions:

Object types and actions

Object	Action	Description
Project	Read	Check the information about the project itself (not including any objects of the project), such as CreateTime.
Project	Write	Update the information of the project itself (not including any objects of the project), such as Comments.
Project	List	View a list of all types of objects in the project.
Project	CreateTable	Create a table in the project.
Project	CreateInstance	Create an instance in the project.
Project	CreateFunction	Create a function in the project.
Project	CreateResource	Create resource in the project.
Project	CreateJob	Create a job in the project.
Project	CreateVolume	Create a volume in the project.
Project	All	All the permissions above.
Table	Describe	Read the metadata of the table.
Table	Select	Read the information of the table.
Table	Alter	Alter the metadata of the table; add or drop partitions.
Table	Update	Override or add data to the table.
Table	Drop	Drop the table.
Table	All	All the preceding permissions.
Function	Read	Read and execute permissions.
Function	Write	Update.
Function	Delete	Delete.
Function	All	All the preceding permissions.
Resource, instance, job, volume	Read	Read permissions.
Resource, instance, job, volume	Write	Update permissions.

Object	Action	Description
Resource, instance, job, volume	Delete	Delete permissions.
Resource, instance, job, volume	All	All the preceding permissions.

Note In the preceding permissions, the CreateTable action of project objects, as well as the Select, Alter, Update, and Drop actions of table objects, must be used together with the CreateInstance action of project objects. Before using the preceding permissions to complete actions, you must assign the CreateInstance permission.

After adding users or creating roles, these users or roles should be authorized. The ACL authorization mechanism of MaxCompute is object-based. Authorization data (the access control list, or ACL) is considered as a sub-resource of an object). Therefore, ACL authorization can be performed only when the objects exist. When the objects are deleted, authorized permission data is automatically deleted.

The ACL of MaxCompute supports authorization using commands like SQL92-defined GRANT/REVOKE commands. Use the corresponding authorization commands to authorize existing project objects or revoke their authorization.

Command syntax:

```
grant actions on object to subject
revoke actions on object from subject
actions ::= action_item1, action_item2, ...
object ::= project project_name | table schema_name | instance inst_name | function func_name | resou
rce res_name
subject ::= user full_username | role role_name
```

Note

The ACL authorization commands of MaxCompute do not support the [WITH GRANT OPTION] parameter. That is, when user A authorizes user B to access an object, user B cannot authorize user C to access the same object. Therefore, all authorization actions must be completed by users with at least one of the following identities:

- Project owner
- Users with the admin role in the project
- Object creators in the project

ACL authorization example:

Scenario: Users alice@aliyun.com and bob@aliyun.com are new members of test_project. In test_project, they must submit jobs, create data tables, and view existing objects of the project. The administrator then takes the following authorization actions:

```
use test_project
-- Open a project.
security
add user aliyun$alice@aliyun.com
-- Add a user.
add user aliyun$bob@aliyun.com
-- Add a user.
create role worker
-- Create a role.
grant worker TO aliyun$alice@aliyun.com
-- Assign a role.
grant worker TO aliyun$bob@aliyun.com
-- Assign a role.
grant CreateInstance, CreateResource, CreateFunction, CreateTable, List ON PROJECT test_project TO
ROLE worke r
-- Authorize a role.
```

1.20.4.5. View permissions

MaxCompute allows you to view permissions in different dimensions. For example, you can view permissions of a specified user, permissions of a specified role, or the authorization list of a specified object.

View the permissions of a user

```
show grants
-- View the access permissions of the current user.
show grants for <username>
-- View the access permissions of a specified user. Only project owners and administrators can view th
e access permissions of a specified user.
```

View the permissions of a role

```
describe role <rolename>
-- View the access permissions granted to a specified role.
```

View the authorization list of an object

```
show acl for <objectName> [on type <objectType>]
-- View the authorization list of a specified object.
```

> **Note** If [on type <objectType>] is not specified, the default type is table.

MaxCompute uses characters A, C, D, and G to indicate the permissions of users or roles. The characters are described as follows:

- A: allow. Access is allowed.
- D: deny. Access is denied.
- C: condition. This is a conditional authorization. This character appears only in the policy authorization system. For more information, see Condition block structure.
- G: grant. You can grant permissions to this object.

Example:

```
odps@test_project> show grants for aliyun$odpstest1@aliyun.com
[roles]
dev
Authorization Type: ACL
[role/dev]
A projects/test_project/tables/t1: Select [user/odpstest1@aliyun.com]
A projects/test_project: CreateTable | CreateInstance | CreateFunction | List
A projects/test_project/tables/t1: Describe | Select
Authorization Type: Policy
[role/dev]
AC projects/test_project/tables/test_*: Describe
DC projects/test_project/tables/alifinance_*: Select [user/odpstest1@aliyun.com]
A projects/test_project: Create* | List
AC projects/test_project/tables/alipay_*: Describe | Select
Authorization Type: ObjectCreator
AG projects/test_project/tables/t6: All
AG projects/test_project/tables/t7: All
```

1.20.5. Cross-project resource sharing

1.20.5.1. Overview

You are the owner or administrator (admin role) of a project, and someone requests to access resources of your project. If the applicant is a member of your project team, we recommend that you use the user and authorization management features for your project. For more information, see User and authorization management of projects. If the applicant is not a member of your project team, you can use the package-based resource sharing feature described in this topic.

A package is used to share data and resources across projects. It can be used to implement cross-project user authorization. The following scenario describes a problem that can only be resolved effectively with the package mechanism.

Members of the Alifinance project need to access Alipay project data. The Alipay project administrator adds Alifinance project users to the Alipay project, and then grants the new users common permissions. For security concerns, the Alipay project administrator does not want to authorize every user of the Alifinance project team. A mechanism is required to allow the Alifinance project administrator to control access to the authorized objects.

By using the package feature, the Alipay project administrator can package the objects that the Alifinance team needs to access, and then allow the package to be installed in the Alifinance project. After installing the package, the Alifinance project administrator can decide whether to grant permissions on the package to users in the Alifinance project.

A package involves two subjects: package creator and package user. The package creator provides resources. The package creator packages the resources to be shared and the corresponding access permissions, and provides the package receiver with the permissions to install and use the package. The package user consumes the resources. After installing the package published by the package creator, the package user can directly access the resources.

The following topics describe the operations that can be performed by a package creator and package user.

1.20.5.2. Package usage

1.20.5.2.1. Operations for package creators

Create a package

Run the following commands to create a package:

```
create package <pkgname>
```

Delete a package

Run the following commands to drop a package:

```
delete package <pkgname>
```

Add a resource to be shared to the package

Run the following commands to add a resource to the package:

```
add project_object to package package_name [with privileges <privileges>]
remove project_object from package package_name
project_object ::= table table_name | instance inst_name | function func_name | resource res_name
privileges ::= action_item1, action_item2, ...
```

Note

- The types of supported objects exclude projects, so you cannot use a package to create objects in other projects.
- In addition to the objects, the operation permissions on the objects are also added to the package. When not passed [with privileges Privileges] When you specify an action permission, the default is read-only, that is, read/describe/select. An object (resource) and its permissions are considered as a whole. You can delete resources in a package. The permissions are revoked when resources are deleted.

Allow other projects to use a package

Run the following commands to allow other projects to use a package:

```
allow project <prjname> to install package <pkgname> [using label <number>]
```

Revoke the permission for other projects to use a package

Run the following commands to revoke another project's permission to use package:

```
disallow project <prjname> to install package <pkgname>
```

View the list of packages already created and installed

Run the following commands to view the list of packages already created and installed:

```
show packages
```

View details of a package

Run the following commands to view details of a package:

```
describe package <pkgname>
```

1.20.5.2.2. Operations for package users

The installed package is a type of independent object in MaxCompute. To access resources in a package (other projects' resources shared with you), you must have the permission to read the package. If you do not have read permissions, submit an application to the project owner or admin for the permissions. The project owner or admin can grant the permissions by using ACL authorization or policy authorization.

For example, the following ACL authorization rule allows user odps_test@aliyun.com to access resources in a package:

```
use prj2 security
install package prj1.testpkg
grant read on package prj1.testpackage to user aliyun$odps_test@aliyun.com
```

The following policy authorization rule allows any user in prj2 to access resources in a package:

```
use prj2
install package prj1.testpkg
put policy /tmp/policy.txt
```

The contents of /tmp/policy.txt are as follow:

```
{
"Version": "1", "Statement": [{
"Effect":"Allow",
"Principal":"*",
"Action":"odps:Read", "Resource":"acs:odps:*:projects/prj2/packages/prj1.testpkg"
}]
}
```

Install a package

Run the following commands to install a package:

```
install package <pkgname>;
```

> **Note** The pkgName of a package to be installed must be in the format of <projectName>.<packageName>.

Uninstall package

Run the following commands to uninstall a package:

```
uninstall package <pkgname>;
```

> **Note** The pkgName of a package to be uninstalled must be in the format of <projectName>.<packageName>.

View packages

Run the following commands to view packages:

```
show packages
-- View the list of packages already created and installed.
describe package <pkgname>
-- View details of a package.
```

1.20.6. Project protection

1.20.6.1. Overview

Some enterprises (such as financial institutions and military enterprises) have high data security requirements. For example, their employees can only perform their jobs in the workplace, and are not allowed to take work materials out of the office. All USB ports on office computers are disabled. These measures aim to prevent leakage of sensitive data.

For example, you are a MaxCompute project administrator in charge of a project with sensitive data. The data must not be shared to other projects. You are required to perform the following configurations to prohibit all operations that could result in data outflow.

1.20.6.2. Data protection

MaxCompute provides a project protection mechanism that prohibits operations that introduce data leakage risks. You can simply configure your project as follows to enable project protection:

```
set security.ProjectProtection=true
-- Enable project protection. This rule allows inbound data flows, but prohibits outbound data flows.
```

After project protection is enabled, the data flow of the project is controlled. Data can flow in, but cannot flow out.

Project protection is disabled by default (ProjectProtection = false). If you have access permissions on multiple projects, you can use any cross-project data access to migrate data between projects. If a project stores highly-sensitive data, the administrator must configure a project protection mechanism.

1.20.6.3. Data export methods when project protection is enabled

After you enable project protection, you may soon encounter this situation: A user (Alice) submits a request to export the data of a table from the project. It is verified that this table contains no sensitive data. MaxCompute provides two methods to export data after project protection is enabled.

Configure an exception policy

The project owner can run the following command to attach an exception policy when enabling project protection:

```
SET ProjectProtection=true WITH EXCEPTION <policyFile>
```

> **Note** This policy mechanism is different from the policy-based authorization mechanism (though the command syntax is the same). This policy describes exceptions of the project protection mechanism. All access requests matching the policy are not subject to the project protection rules.

Example:

The following policy allows the user Alice@aliyun.com to export data out of the alipay project when performing the SELECT operation on the alipay.table_test table in a SQL task:

```
{
"Version": "1", "Statement": [{
"Effect":"Allow", "Principal":"ALIYUN$Alice@aliyun.com",
"Action":["odps:Select"],
"Resource":"acs:odps:*:projects/alipay/tables/table_test",
"Condition":{
"StringEquals": { "odps:TaskType":["DT", "SQL"]
}
}
}]
}
```

Note

- The preceding exception policy does not grant any permissions. If Alice does not have the SELECT permission on alipay.table_test, the preceding exception policy does not allow Alice to export data. Project protection specifies data flow control, not access control. Data flow control is effective only when a user can access the target data.
- Data leakage due to TOC2TOU (also known as the race condition problem) arises in the following situation:
  i. [TOC stage] User A submits an application to the project owner to export table t1. The project owner verifies that t1 does not contain sensitive data. The project owner configures an exception policy, which allows user A to export t1.
  ii. A malicious user changes the content of t1 by writing sensitive data to it.
  iii. [TOU stage] User A exports t1. The t1 exported by the user is not the same t1 that was authorized by the project owner.

  Suggestions on TOC2TOU prevention: For a table that a user applies to export, the project owner must make sure that no other user (including admins) can update the table or create a table with the same name (using DROP TABLE and CREAT TABLE). In the preceding TOC2TOU scenario, we recommend that the project owner create a snapshot of t1 in step 1. Then, create an exception policy for the user to use this snapshot. Do not grant the admin role to any users.

Configure a trusted project

If the current project is protected, and the target project is a trusted project of the current project, data flows to the target project are not subject to the project protection rules. If each project in a group is mutually configured as trusted projects, the group is considered a trusted project group. Data can flow freely within the group, but cannot flow out.

Run the following command to manage trusted projects:

```
ist trustedprojects
-- Show all trusted projects of the current project.
add trustedproject <projectname>
-- Add a trusted project of the current project.
remove trustedproject <projectname>
-- Remove a trusted project of the current project.
```

1.20.6.4. Resource sharing and data protection

In MaxCompute, package-based resource sharing and project protection are mutually independent mechanisms that take effect at the same time, but their functions are mutually restrictive.

In MaxCompute, resource sharing takes precedence over project protection. If a data object is shared to users in other projects through resource sharing, the project protection rules will not apply to this data object.

To prevent data outflow from the project, after you enable project protection (ProjectProtection=true), you must verify the following points:

- Make sure that no trusted projects are added. If one is added, evaluate possible risks.
- Make sure that no exception policies are configured. If one is configured, evaluate possible risks, especially risks due to TOC2TOU.
- Check whether package data sharing is not in use. If package data sharing is in use, make sure that the package contains no sensitive data.

1.20.7. Project security configuration

MaxCompute is a multi-tenant data processing platform. Different tenants may have different data security requirements. MaxCompute provides project-level security configuration to satisfy data security requirements of different tenants. Project owners can customize their external accounts and authentication models as required.

MaxCompute supports multiple orthogonal authorization mechanisms, such as ACL-based authorization, policy-based authorization, and implicit authorization (for example, an object creator is automatically authorized to access the object). However, not all users require these security mechanisms. You can configure an authentication model that best suits your business demands and usage habits.

```
show SecurityConfiguration
-- View the security configuration of the project.
set security.CheckPermissionUsingACL=true/false
-- Enable or disable ACL-based authorization. Default value: true.
set security.CheckPermissionUsingPolicy=true/false
-- Enable or disable policy-based authorization. Default value: true.
set security.ObjectCreatorHasAccessPermission=true/false
-- Allow or disallow an object creator to be granted the object access permission by default. Default v
alue: true.
set security.ObjectCreatorHasGrantPermission=true/false
-- Allow or disallow an object creator to be granted the authorization permission by default. Default v
alue: true.
set security.LabelSecurity=true/false
-- Enable or disable the label security policy.
set security.ProjectProtection=true/false
-- Enable or disable project protection to allow or prohibit data transfer from the project.
```

1.20.8. Authorization policies

1.20.8.1. Policy overview

Policy authorization is a principal-based authorization. Permission data authorized by policy (that is, access policy) is considered as a type of sub-resource of the authorization subject. Policy authorization can be performed only if the subject exists. When a subject is deleted, their authorization data is deleted automatically. Policy authorization uses an access policy language customized for MaxCompute to allow or deny subjects access to project objects.

Policy authorization is a new authorization mechanism mainly used to handle complicated authorization scenarios that ACL authorization struggles to deal with, such as:

- Authorize a group of objects, such as all functions and all tables starting with taobao, at a time.
- For authorizations with restrictive conditions, such as one that takes effect only in a specified period, one that takes effect only if the requester initiates the request from a specified IP addresses, or one that allows the user to use SQL only (disallowing other tasks) to access a table.

The command format of policy authorization is as follows:

```
GET POLICY;
-- Read the project policy.
PUT POLICY <policyFile>;
-- Set (overwrite) the project policy.
GET POLICY ON ROLE <roleName>;
-- Read the policy of a role in the project.
PUT POLICY <policyFile> ON ROLE <roleName>;
-- Set (overwrite) the policy of a role in the project.
```

> **Note** MaxCompute currently supports project policies and role policies. A project policy applies to all users of the project, while a role policy applies only to users to whom the role is assigned. You must specify a principal (user) for project policies, but you cannot specify a principal for role policies, because the role will be assigned to users.

An example of project policy authorization is as follows:

Scenario: Authorized user alice@aliyun.com can only submit a request before 23:59:59 2017-11-11 from an IP address in the subnet 10.32.180.0-23, and can only perform the CreateInstance, CreateTable, and List operations in test_project. No tables in test_project can be dropped.

The policy is as follows:

```
{
"Version": "1", "Statement": [{
"Effect":"Allow", "Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:CreateTable","odps:CreateInstance","odps:List"],
"Resource":"acs:odps:*:projects/test_project",
"Condition":{ "DateLessThan": {
"acs:CurrentTime":"2017-11-11T23:59:59Z"
},
"IpAddress": { "acs:SourceIp":"10.32.180.0/23"
}
}
},
{
"Effect":"Deny", "Principal":"ALIYUN$alice@aliyun.com", "Action":"odps:Drop", "Resource":"acs:odps:
*:projects/test_project/tables/*"
}]
}
```json
```
```

```json
{
"Version": "1", "Statement": [{
"Effect":"Allow", "Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:CreateTable","odps:CreateInstance","odps:List"],
"Resource":"acs:odps:*:projects/test_project",
"Condition":{ "DateLessThan": {
"acs:CurrentTime":"2017-11-11T23:59:59Z"
},
"IpAddress": { "acs:SourceIp":"10.32.180.0/23"
}
}
},
{
"Effect":"Deny", "Principal":"ALIYUN$alice@aliyun.com",
"Action":"odps:Drop",
"Resource":"acs:odps:*:projects/test_project/tables/*"
}]
}
```

```
{
"Version": "1", "Statement": [{
"Effect":"Allow", "Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:CreateTable","odps:CreateInstance","odps:List"],
"Resource":"acs:odps:*:projects/test_project",
"Condition":{ "DateLessThan": {
"acs:CurrentTime":"2017-11-11T23:59:59Z"
},
"IpAddress": { "acs:SourceIp":"10.32.180.0/23"
}
}
},
{
"Effect":"Deny", "Principal":"ALIYUN$alice@aliyun.com",
"Action":"odps:Drop",
"Resource":"acs:odps:*:projects/test_project/tables/*"
}]
}
```

Note

- Currently, only role policies and project policies are supported, and user policies are not.
- Every policy only supports one policy file. Since put policies override existing policies, you must follow the sequence below to modify a policy:
  i. Get Policy.
  ii. Merge policy statements.
  iii. Put policies.

1.20.8.2. Policy-related terms

Permission is a basic concept of access control. When a requester wants to take an action on a resource, the action may be allowed or denied, based on the permission settings. A statement refers to the formal description of a single permission, and policy refers to a set of statements.

An access policy comprises the following access control elements: principal, action, resource, access restriction, and effect. These elements are briefly described below.

Principal

A principal of an object is a user or group to which permissions are assigned in an access policy. For example, the access policy allows Michael to perform the CreateObject action on the resource SampleBucket before December 31, 2017. Michael is the principal of the object.

Action

An action is an activity that the principal has permission to perform. For example, the access policy allows Michael to perform the CreateObject action on the resource SampleBucket before December 31, 2017. Therefore, CreateObject is an action of the access policy.

Resource

Resource is the object a principal requests access to. For example, the access policy allows Michael to perform the CreateObject action on the resource SampleBucket before December 31, 2017. SampleBucket is a resource of the access policy.

Access restriction

Access restriction is the prerequisite for the permission to take effect. For example, the access policy allows Michael to perform the CreateObject action on resource SampleBucket before December 31, 2017. The access restriction is before December 31, 2017.

Effect

Authorization effect has two options: Allow (action) or deny (action). In general, deny actions are generally more efficient and are checked first during permission checks.

Notice The deny action and revoking pemission are completely different concepts. The latter usually revokes permissions for both allow action and deny action. For example, a traditional database supports the revoke and revoke deny actions.

1.20.8.3. Access policy structure

1.20.8.3.1. Overview

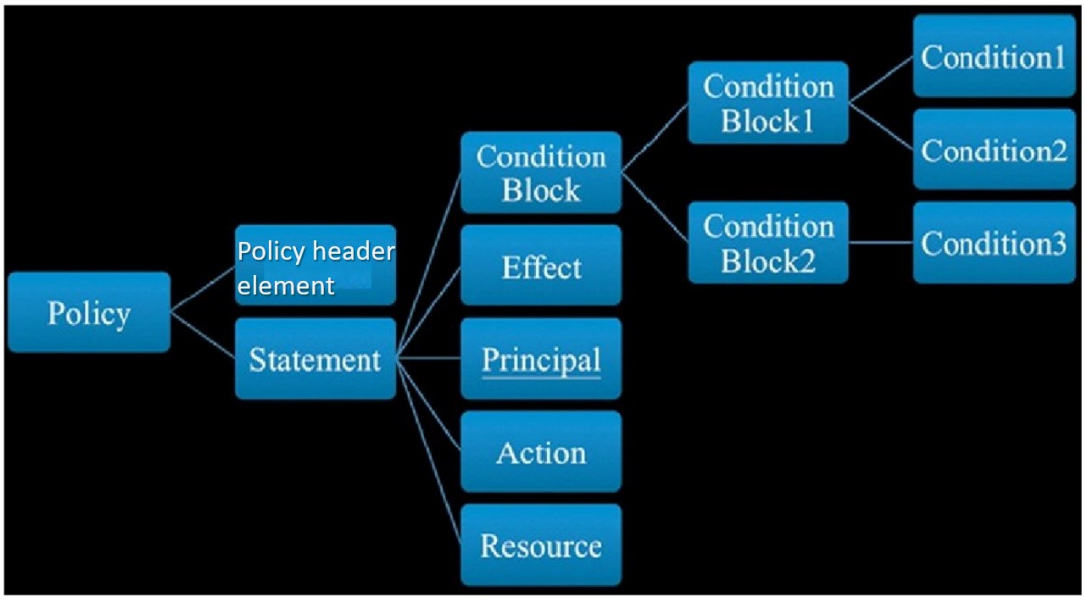
The following figure shows the structure of an access policy. A policy consists of the following parts:

- An optional policy header
- One or more statements

The policy header is optional and includes the policy version. The policy body is a set of statements.

The following figure shows the structure of a policy.

Policy structure



1.20.8.3.2. Authorization statement structure

An authorization statement includes the following entries:

- Effect: indicates the permission type of this statement. The value can be either Allow or Deny.
- Principal: If a policy is bound to a user or role in the authorization process, such as the role policy of MaxCompute, you cannot appoint a principal. If a policy is bound to a project or objects of the project in the authorization process, such as the project policy of MaxCompute, you must specify a principal.
- Action: indicates the authorization operation. It can be one or more operation names, and supports the asterisk (*) and question mark (?) wildcard characters. The asterisk (*) matches any number of characters, and the question mark (?) matches a single character. For example, Action = * indicates all operations.
- Resource: indicates the authorization object. It can be one or more object names, and supports the asterisk (*) and question mark (?) wildcard characters. The asterisk (*) matches any number of characters, and the question mark (?) matches a single character. For example, Resource = * indicates all objects.
- Condition block: indicates the conditions that must be met for the permission described by this

authorization statement to take effect. See the next topic for the structure of the condition block.

1.20.8.3.3. Conditional block structure

A condition block consists of one or more condition clauses. A condition clause consists of an action type, keyword, and condition value. The action types and keywords will be described in detail in the subsequent sections.

Whether a condition block is satisfied is determined as follows:

- A conditional keyword can correspond to one or more values. If the conditional keyword value is equal to one of the corresponding values, the condition is satisfied.
- A condition clause of a conditional operation type is satisfied if all conditional keywords in the clause are satisfied.
- A condition block is satisfied only if all of its condition clauses are satisfied.

1.20.8.3.4. Conditional action type

The following action types are supported: string, number, date, Boolean, and IP address. The methods supported by each conditional operation type are as follows:

String:

```
StringEquals
StringNotEquals
StringEqualsIgnoreCase
StringNotEqualsIgnoreCase
StringLike
StringNotLike
```

Numeric:

```
NumericEquals
NumericEquals
NumericLessThan
NumericLessThanEquals
NumericGreaterThan
NumericGreaterThanEquals
```

Date and time:

```
DateEquals
DateNotEquals
DateLessThan
DateLessThanEquals
DateGreaterThan
DateGreaterThanEquals
```

Boolean:

```
Bool
```

IP address:

```
IpAddress NotIpAddress
```

1.20.8.3.5. Conditional keywords

MaxCompute supports the conditional keywords reserved by Alibaba Cloud Service (ACS). The following table describes these conditional keywords.

Conditional keywords

Conditional keywords reserved by ACS	Type	Description
acs:CurrentTime	Date and time	The time when the Web server receives a request. It is based on the ISO 8601 standard, for example, 2017-11-11T23:59:59Z.
acs:SecureTransport	Boolean	Whether the request is sent over a secure channel, such as an HTTPS channel.
acs:SourceIp	IP address	The IP address of the client that sent the request.
acs:UserAgent	String	The UserAgent of the client that sent the request.
acs:Referer	String	The HTTP referer that sent the request.

Note acs:SourceIp refers to the remote_ip of the HTTP connection, not the (leftmost) client IP address in the x-forwarded-for HTTP header field. For example, if 10.230.205.105 is a LAN IP address, acs:SourceIp is the egress gateway IP address of this LAN. If the network egress uses a proxy server, acs:SourceIp is the IP address of the proxy server. If the request traverses across multiple proxy servers, acs:SourceIp is the IP address of the final proxy server. The value of acs:SourceIp may vary depending on the rules configured on the proxy server.

1.20.8.4. Access policy norm

1.20.8.4.1. Principal naming convention

The principal is the request sender. Currently, only an Alibaba Cloud account, domain account, or Taobao account is accepted as a principal. A cloud account can be represented by ID or DisplayName.

Example:

```
"Principal": "43274"
"Principal": "ALIYUN$bob@aliyun.com"
"Principal": ["ALIYUN$bob@aliyun.com","ALIYUN$jack@aliyun.com", "TAOBAO$alice"]
```

1.20.8.4.2. Resource naming convention

The following naming conventions are used for MaxCompute resources.

```
acs:<service-name>:<namespace>:<relative-id>
```

The parameters are described as follows.

Parameters

Name	Description
acs	Retained resource header.
service-name	The name of an open cloud service, such as MaxCompute, OSS, and TableStore.
namespace	Naming space, used for resource isolation. If a cloud account ID is used for resource isolation, the value can be the cloud account ID. If this option is not supported, use the asterisk (*) wildcard character instead.
relative-id	Indicates the service-related resource. Its meaning depends on specific services. This part of the format description supports a tree structure similar to the file path. Using MaxCompute as an example, the format of relative-id is: `projects/<project_name>/<object_type>/<object_name>`

Some MaxCompute resource naming examples.

Naming examples

Item	Description

Item	Description
*	All objects in the project.
projects/prj1/tables/t1	Table t1 of Project prj1.
projects/prj1/instances/*	All instances of Project prj1.
projects/prj1/tables/*	All tables of Project prj1.
projects/prj1/tables/taobao*	All tables of Project prj1 whose names start with "taobao".

1.20.8.4.3. Action naming

Action naming conventions are as follows:

```
<service-name>:<action-name>
```

Description:

- service-name: name of an open cloud service, for example, maxcompute, oss, and table store.
- action-name: name of service-related action APIs.

The following table lists MaxCompute action naming examples.

MaxCompute action naming examples

Naming example	Description
*	All actions.
odps:*	All MaxCompute actions.
odps:CreateTable	The CreateTable action of MaxCompute.
odps:Create*	All MaxCompute actions whose names start with "Create".

1.20.8.4.4. Condition keys naming

The naming format for condition keys retained by the open cloud service is:

```
acs:<condition-key>
```

Description:

condition-key: ACS reservs 5 types of condition keys, which are accessible for all open services. They are: acs:CurrentTime, acs:SecureTransport, acs:SourceIp, acs:UserAgent, acs:Referer.

The naming format for condition keys related to the specific service is:

```
<service-name>:<condition-key>
```

Description:

Condition-key: service-defined condition key.

1.20.8.4.5. Access policy example

Policy example:

```
{
"Version": "1",
"Statement": [{
"Effect":"Allow",
"Principal":"ALIYUN$alice@aliyun.com",
"Action":["odps:CreateTable","odps:CreateInstance","odps:List"],
"Resource":"acs:odps:*:projects/prj1",
"Condition":{ "DateLessThan": {
"acs:CurrentTime":"2017-11-11T23:59:59Z"
},
"IpAddress": { "acs:SourceIp":"10.32.180.0/23"
}
}
},
{
"Effect":"Deny", "Principal":"ALIYUN$alice@aliyun.com",
"Action":"odps:Drop",
"Resource":"acs:odps:*:projects/prj1/tables/*"
}]
}
```

Note The authorized user (alice@aliyun.com) can only submit a request from subnet 10.32.180.0/23 before 2017-11-11T23:59:59Z. The user can only perform the CreateInstance, CreateTable, and List operations on the prj1 project. The user cannot delete tables from prj1.

1.20.8.5. Differences between policy authorization and ACL authorization

ACL authorization:

- Use ACL authorization to grant or revoke permissions when both the grantee (such as a user or role) and the object (such as a table) exist. Like the security feature of Oracle authorization, this avoids the security risk out of dropping and recreating an object with the same name.
- When dropping an object, all authorizations related to the object are automatically revoked.
- It only supports allow (whitelist) authorization, and does not support deny (blacklist)

authorization.

- Use the classic Grant/Revoke commands for authorization. The command is simple and not prone to mistakes. Conditional authorization is not supported.
- This method is suitable for simple scenarios where condition or deny is not needed for authorization, and only the existing objects need to be authorized.

Policy authorization:

- Use policy authorization to grant or revoke permissions when the grantee or object is not available. The Object parameter supports wildcard "". For example, *projects/tbproj/tables/taobao* matches all tables whose names start with taobao in project tbproj. Like the features of MySQL authorization, policy authorization allows a non-existent object to be authorized, so the authorizer must consider the security risks of dropping and recreating an object with the same name.
- When dropping an object, the policy authorization related to this object is not deleted.
- Both allow (whitelist) authorization and deny (blacklist) authorization are supported. If allow and deny conflict, the deny action takes priority.
- Conditional authorization is supported. The authorizer can enforce 20 conditions on allow or deny authorization. For example, these conditions can be used to limit access to IP addresses within a subnet, and allow access before 23:59:59 on November 11, 2017.
- This method is suitable for relatively complicated scenarios where conditional authorization and deny action are needed and non-existent objects need to be authorized.
- Using policy authorization is more complicated than ACL authorization, but provides more flexibility.

1.20.8.6. Application limits

Application limits

Item	Limit	Description
ACCESS_POLICY_SIZE_LIMIT	32 KB	The maximum size of AccessPolicy.
USER_NUMBER_LIMIT_IN_ON E_PROJECT	1,000	The maximum number of users in a project.
ROLE_NUMBER_LIMIT_IN_ON E_PROJECT	500	The maximum number of roles in a project.
ROLE_NAME_LENGTH_LIMIT	64	The maximum number of characters in a role name.
SECURITY_COMMENT_SIZE_L IMIT	1 KB	The maximum size of a security comment.
PACKAGE_NAME_LENGTH_LI MIT	128	The maximum number of characters in a package name.
ALLOW_PROJECT_NUMBER_LIMIT_IN_ONE_P ACKAGE	1024	The maximum number of projects installed in a package.

Item	Limit	Description
RESOURCE_NUMBER_LIMIT_IN_ONE_PACKAGE	256	The maximum number of resources in a package.
PACKAGE_NUMBER_LIMIT_IN_ONE_PROJECT	512	The maximum number of packages that can be created in a project.
INSTALLED_PACKAGE_NUMBER_LIMIT_IN_ONE_PROJECT	64	The maximum number of packages that can be installed in a project.

1.20.9. Collection of security statements

1.20.9.1. Project security configuration

Authentication

Authorization configuration statements

Statement	Description
show SecurityConfiguration	Displays the project security configuration.
set CheckPermissionUsingACL=true/false	Enables or disables ACL-based authorization.
set CheckPermissionUsingPolicy=true/fal se	Enables or disables policy-based authorization.
set ObjectCreatorHasAccessPermission=true/false	Allows or disallows an object creator to be granted the object access permission by default.
set ObjectCreatorHasGrantPermission=true/false	Allows or disallows an object creator to be granted the ACL-based authorization permission by default.

Data protection

Data protection statements

Statement	Description
set ProjectProtection=false	Disables project protection.
set ProjectProtection=true [with exception <policy>]	Enables project protection.
list TrustedProjects	Displays the list of trusted projects.
add TrustedProject <projectName>	Adds a project to trusted projects.

Statement	Description
remove trustedproject <projectname>;	Removes a project from trusted projects.

1.20.9.2. Project permission management

User management

User management statements

Statement	Description
list users	Lists all added users.
add user <username>	Adds a user.
remove user <username>	Removes a user.

Role management

Role management statements

Statement	Description
list roles	Lists all created roles.
create role <rolename>	Creates a role.
drop role <rolename>	Deletes a role.
grant <rolelist> to <username>	Revokes roles from a user.
revoke <rolelist> from <username>	Grants one or more roles to a user.

ACL-based authorization

ACL-based authorization statements

Statement	Description
grant <privList> on <objType> <objName> to user <username>	Authorizes a user.
grant <privList> on <objType> <objName> to role <rolename>	Authorizes a role.
revoke <privList> on <objType> <objName> from user <username>	Revokes permissions from a user.

Statement	Description
revoke <privList> on <objType> <objName> from role <rolename>	Revokes permissions from a role.

Policy-based authorization

Policy-based authorization statements

Statement	Description
get policy	Displays policy settings at the project level.
put policy <policyFile>	Configures a policy at the project level.
get policy on role <roleName>	Displays the policy settings of a role.
put policy <policyFile> on role <roleName>	Configures a policy for a role.

Permission review

Permission review statements

Statement	Description
whoami	Displays information about the current user.
show grants [for <username>] [on type <objectType>]	Displays permissions and roles of a user.
show acl for <objectName> [on type <objectType>]	Displays the authorization information of an object.
describe role <roleName>	Displays the authorization and assignment information of a role.

1.20.9.3. Package-based resource sharing

Resource sharing

Resource sharing statements

Statement	Description
create package <pkgName>	Creates a package.
delete package <pkgName>	Deletes a package.
add <objType> <objName> to package<pkgName> [with privileges privs]	Adds resources that need to be shared to a package.

Statement	Description
remove <objType> <objName> from package <pkgName>	Removes shared resources from a package.
allow project <prjName> to install package <pkgName> [using label <num>]	Allows a project to use a user package.
disallow project <prjName> to install package <pkgName>	Disables a project from using a user package.

Resource use

Resource use statements

Statement	Description
install package <pkgName>	Installs a package.
uninstall package <pkgName>	Uninstalls a package.

Package viewing

Package viewing statements

Statement	Description
show packages	Lists all created and installed packages.
describe package <pkgName>	Views details of a package.

1.21. Frequently-used tools

1.21.1. MaxCompute console

1.21.1.1. Usage notes

This topic provides usage notes for the MaxCompute client.

> **Notice**
>
> - Do not rely on the output data of the client in any development or planning processes, as the data format may change. The client output format may not be forward compatible. The command syntax and behavior vary according to versions.
> - The MaxCompute client is a Java program. It requires JRE to run. You need to download and install JRE 1.8 to use the MaxCompute client.

For more information about how to configure and use the client, see Quick start.

1.21.1.2. Install the client

This topic describes how to install the MaxCompute client.

1. Download the client package to your client computer.
2. Decompress the client package to a folder, where you can see the following folders:

```
bin/
conf/
lib/
plugins/
```

3. Edit the following parameters in the odps_config.ini file in the conf folder:

```
project_name=
access_id=**********
access_key=**********
end_point= <MaxCompute service address>
```

> **Note**
> - Set access_id and access_key to the AccessKey ID and AccessKey Secret of your cloud account.
> - If you frequently use a project, enter the project name after project_name=. Then, you do not need to run the `use project_name;` command each time you log on to the client.

4. After the modifications, run the odps file in the bin directory (*./bin/odpscmd* in a Linux system or *./bin/odpscmd.bat* in a Windows system). Then, you are ready to run SQL statements. An example is as follows:

```
create table tbl1(id bigint);
insert overwrite table tbl1 select count(*) from tbl1;
select 'welcome to MaxCompute!' from tbl1;
```

1.21.1.3. Configuration description

This topic describes some configurations and corresponding parameters of the MaxCompute client.

Help

Run the following command to view the help information about the client:

```
odps@ >./bin/odpscmd -h;
```

> **Note** You can also enter `h;` or `help;` (case insensitive) in the interaction mode.

Startup parameters

Run the following command to specify a number of startup parameters:

```
Usage: odpscmd [OPTION]...
where options include:
  --help (-h)for help
  --project= use project
  --endpoint= set endpoint
  -u  -p  user name and password
  -k  will skip beginning queries and start from specified position
  -r  set retry times
  -f <"file_path;"> execute command in file
  -e <"command;[command;]..."> execute command, include sql command
  -C will display job counters
```

Example: (-f is used as an example)

1. Prepare a local script file named script.txt. The file is stored in D:/. Its contents are as follows:

   ```
   DROP TABLE IF EXISTS test_table_mj;
   CREATE TABLE test_table_mj (id string, name string);
   DROP TABLE test_table_mj;
   ```

2. Run the following command:

   ```
   odpscmd\bin>odpscmd -f d:/script.txt;
   ```

3. The command output is as follows:

```
ID = 20170528122432906gux77io3
Log view:
http://webconsole.odps.aliyun-inc.com:8080/logview/?h=http://service-corp.odps.aliyun- inc.com
/api&podps_public_dev&i2017052 8122432906gux77io3&tokenRnlrSzJoL242YW43dFFIc1dmb1ZWZzFx
Q1RFPSxPRFBTX09CTzoxMDcwMDI1NjI3ODA1 NjI5LDE0MzM0MjA2NzMseyJTdGF0ZW1lbnQiOlt7IkFjdGl
vbiI6WyJvZHBzOljlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB yb2
plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0MzI5MDZndXg3N2lvMyJdfV0sIlZl
cnNpb24iOiIxIn0=
OK
ID = 20170528122439318gcmkk6u1
Log view:
http://webconsole.odps.aliyun-inc.com:8080/logview/?h=http://service-corp.odps.aliyun- inc.com
/api&podps_public_dev&i2017052 8122439318gcmkk6u1&tokendSt0RXdlV0M5YjZET2I1MnJuUFkzWDN
1aWpzPSxPRFBTX09CTzoxMDcwMDI1NjI3ODA1NjI5LDE0MzM0MjA2ODAseyJTdGF0ZW1lbnQiOlt7IkFjd
GlvbiI6WyJvZHBzOljlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB y
b2plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0MzkzMThnY21razZ1MSJdfV0s
IlZlcnNpb24iOiIxIn0=
OK
ID = 20170528122440389g98cmlmf
Log view:
http://webconsole.odps.aliyun-inc.com:8080/logview/?h=http://service-corp.odps.aliyun- inc.com
/api&podps_public_dev&i2017052 8122440389g98cmlmf&tokenNWlwL0EvQThxUXhzcTRERDc5NFg0b2
IxZ3QwPSxPRFBTX09CTzoxMDcwMDI1NjI3OD A1NjI5LDE0MzM0MjA2ODAseyJTdGF0ZW1lbnQiOlt7IkFj
dGlvbiI6WyJvZHBzOljlYWQiXSwiRWZmZWN0IjoiQWxsb3ciLCJSZXNvdXJjZSI6WyJhY3M6b2RwczoqOnB
yb2plY3RzL29kcHNfcHVibGljX2Rld i9pbnN0YW5jZXMvMjAxNTA1MjgxMjI0NDAzODlnOThjbWxtZiJdfV0s
IlZlcnNpb24iOiIxIn0=
OK
```

Interactive mode

Directly run the client, and you will enter the interaction mode. The following information is displayed:

```
[admin: ~]$odpscmd
Aliyun ODPS Command Line Tool
Version 1.0
@Copyright 2012 Alibaba Cloud Computing Co., Ltd. All rights reserved.
XXX@ XXX> INSERT OVERWRITE TABLE DUAL SELECT * FROM DUAL;
```

> **Note** The first XXX indicates the identifier of MaxCompute, and the second XXX indicates the project to which you belong. Enter a command at the cursor (terminated by a semicolon), and press Enter to run it.

Command output

The output of a SQL statement is in either HumanReadable (default) or MachineReadable format. If you use the -M parameter when running odpscmd, the output format is CSV.

> **Note** This function currently applies only to SELECT and READ statements, and takes effect when reading data.

Continue run

When you run odpscmd in -e or -f mode and want to start with an intermediate statement among a few statements, you can use the -k parameter. This parameter indicates that the execution starts from the specified statement while the preceding statements are skipped. If the parameter value is equal to or smaller than 0, execution starts from the first statement. A statement terminated by a semi-colon is considered a valid statement. At runtime, the process indicates the specific statement that is running successfully or fails.

For example, the */tmp/dual.sql* file includes the following three SQL statements:

```
drop table dual;
create table dual (dummy string);
insert overwrite table dual select count(*) from dual;
```

You can run the following command to ignore the first two statements:

```
odpscmd-k 3 -f dual.sql
```

Obtain information about the current logon user

Run the following command to obtain the cloud account of the current logon user and the endpoint that is used:

```
whoami
```

Example:

```
odps@ hiveut>whoami; Name: odpstest@aliyun.com ID: 1090142773636588 End_Point: <MaxCompute s
ervice address> Project: lijunsecuritytest
```

Exit

Command syntax:

```
odps@ > quit;
```

or

```
q;
```

Configure a job priority

Command syntax:

```
Admin@ > ./bin/odpscmd --instance-priority=<PRIORITY>;
```

Configuration file: odps_config.ini

```
instance_priority=<PRIORITY>
```

> **Notice**
>
> - The value range of <PRIORITY > is 0-9, where 0 indicates the highest priority and 9 the lowest priority.
> - The priority setting in the configuration file applies to all instances submitted in CLT.
> - The priority value is 9 by default.

DryRun mode

In the DryRun mode, MaxCompute parses a SQL statement to check if the syntax is correct and generates an execution plan. MaxCompute does not submit a distributed job. Command syntax:

```
./bin/odpscmd -y
```

SQL reliability

If an exception is returned during the execution of SQL statements such as INSERT or CREATE TABLE AS, the client tries to automatically recover data and metadata to the pre-execution state based on known information.

- Data that is overwritten by INSERT OVERWRITE during QUERY execution is recovered from the temporary backup directory to the original directory.
- Data that is generated by INSERT INTO during QUERY execution is removed.
- Tables created during QUERY execution and dynamically generated partition information are removed.

If MaxCompute fails to recover data, it returns a specific error code. The error code alerts you that further attempts can make some data unrecoverable. The error code is as follows:

```
ODPS-
0110999: Critical! Internal error happened in commit operation and rollback failed, possible breach of a
tomicity
```

1.21.2. Eclipse development plugin

1.21.2.1. Install Eclipse

This topic describes how to install the Eclipse plug-in.

Context

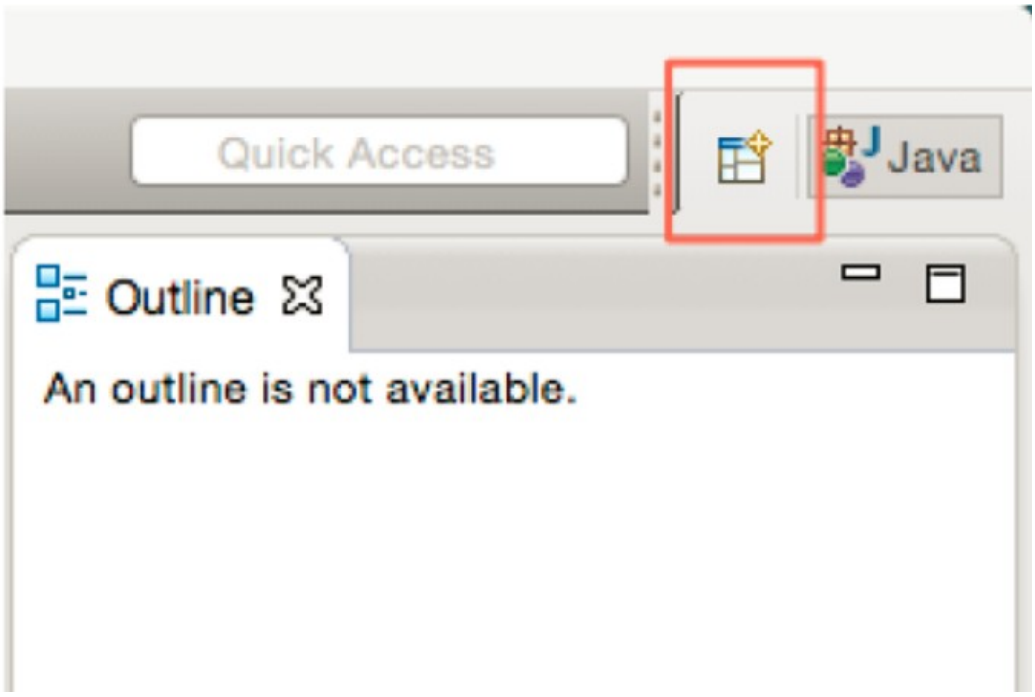
MaxCompute provides the Eclipse plug-in to help you easily use the Java SDKs for MapReduce and UDFs in your development work. This plug-in can simulate the running process of MapReduce or UDFs. It provides local debugging methods and simple template generation features. You can download the software package from Eclipse.

> **Note** Unlike the local running mode of MapReduce, the Eclipse plug-in cannot synchronize data with MaxCompute. You need to manually copy the required data to the warehouse directory of Eclipse.

Procedure

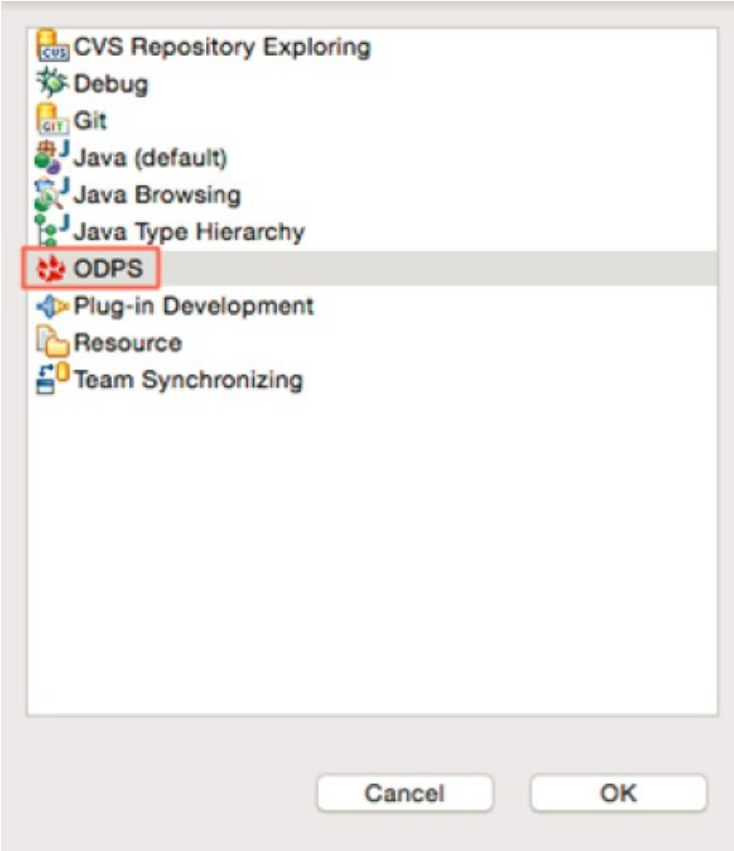
1. Decompress the Eclipse package to obtain the following JAR file: *odps-eclipse-plugin-bundle-0.15.0.jar*
2. Place the JAR file in the plugins folder under the Eclipse installation directory.
3. Start Eclipse and click the Open Perspective icon in the upper-right corner, as shown in the following figure.

   Eclipse installation 1

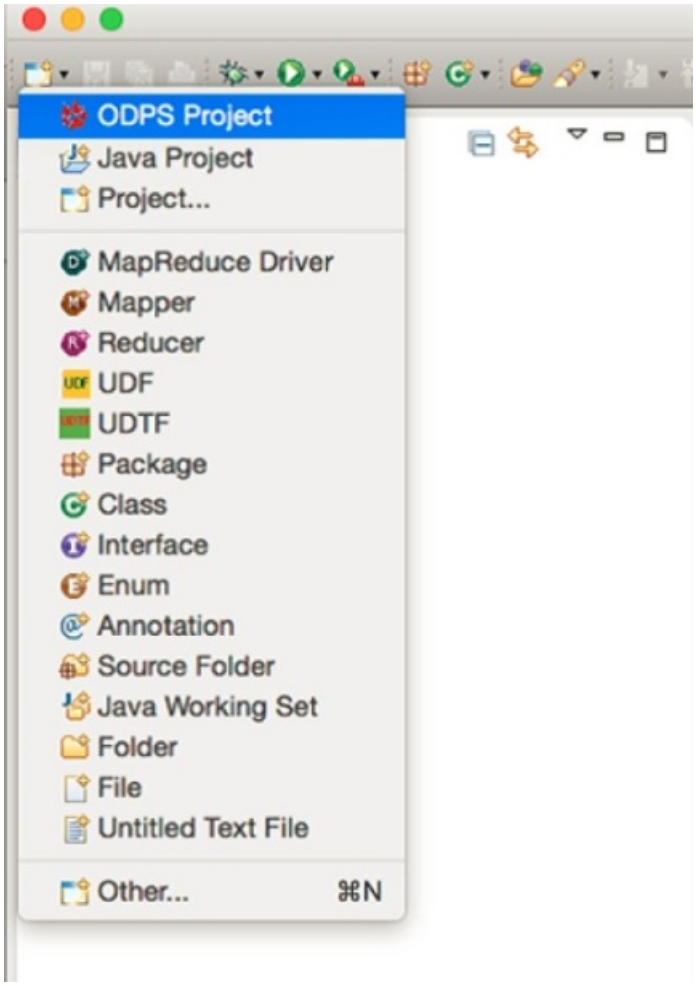
4. Click **ODPS** in the dialog box that appears, as shown in the following figure.

   Eclipse installation 2



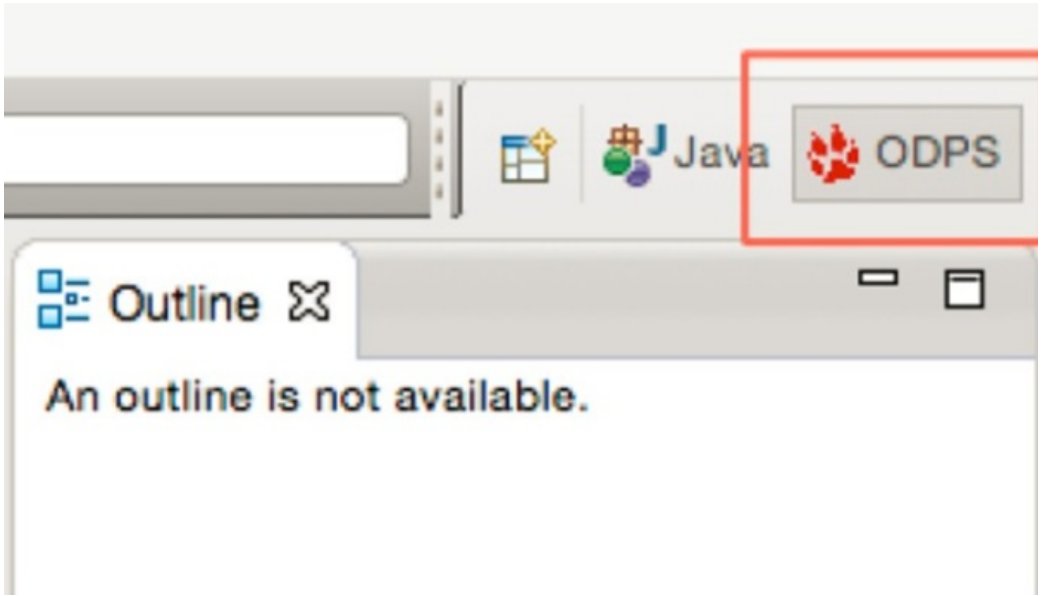
5. After you click **ODPS**, a navigation pane appears, as shown in the following figure.

   Eclipse installation 3



6. Click **ODPS Project** in the navigation pane and then click **OK**. The ODPS icon is displayed in the upper-right corner, as shown in the following figure. The icon indicates that the plug-in has taken effect.

Eclipse installation 4



1.21.2.2. Create a project

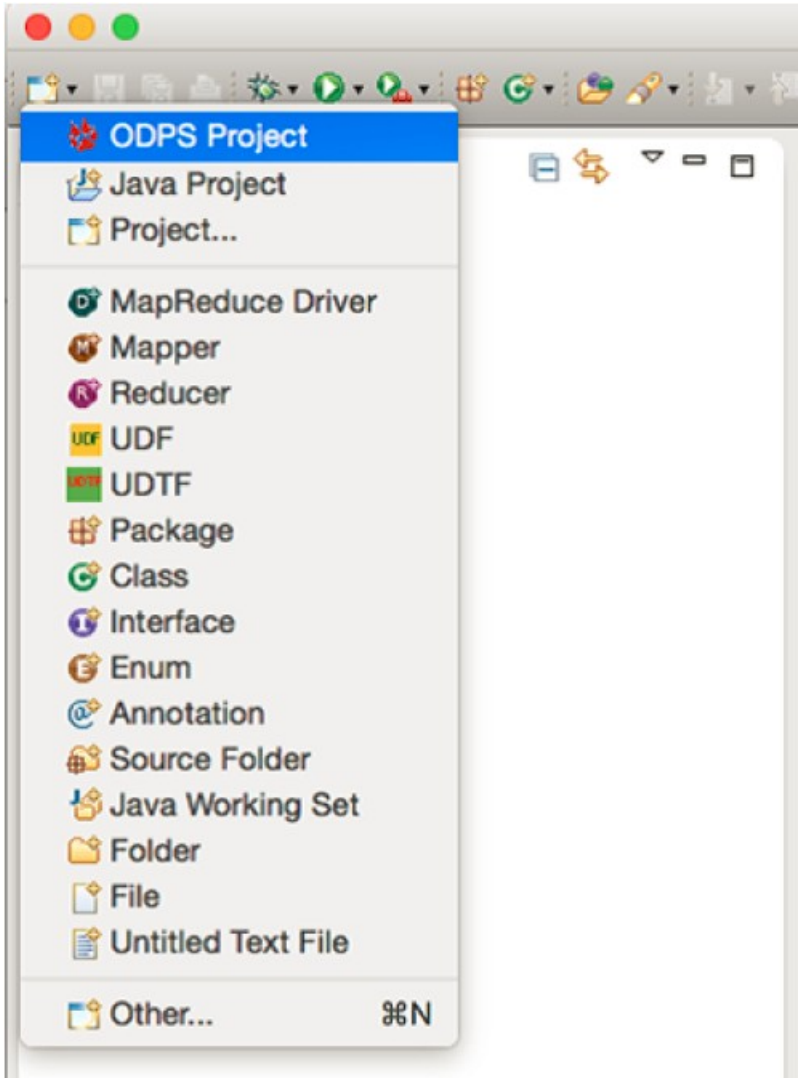
1.21.2.2.1. Method 1

This topic describes the first method to create a project in the Eclipse development plugin.

Procedure

1. Start Eclipse. Choose **File** > **New** > **Project** > **ODPS** > **ODPS Project** in the upper-left corner to create a project, as shown in the following figure.

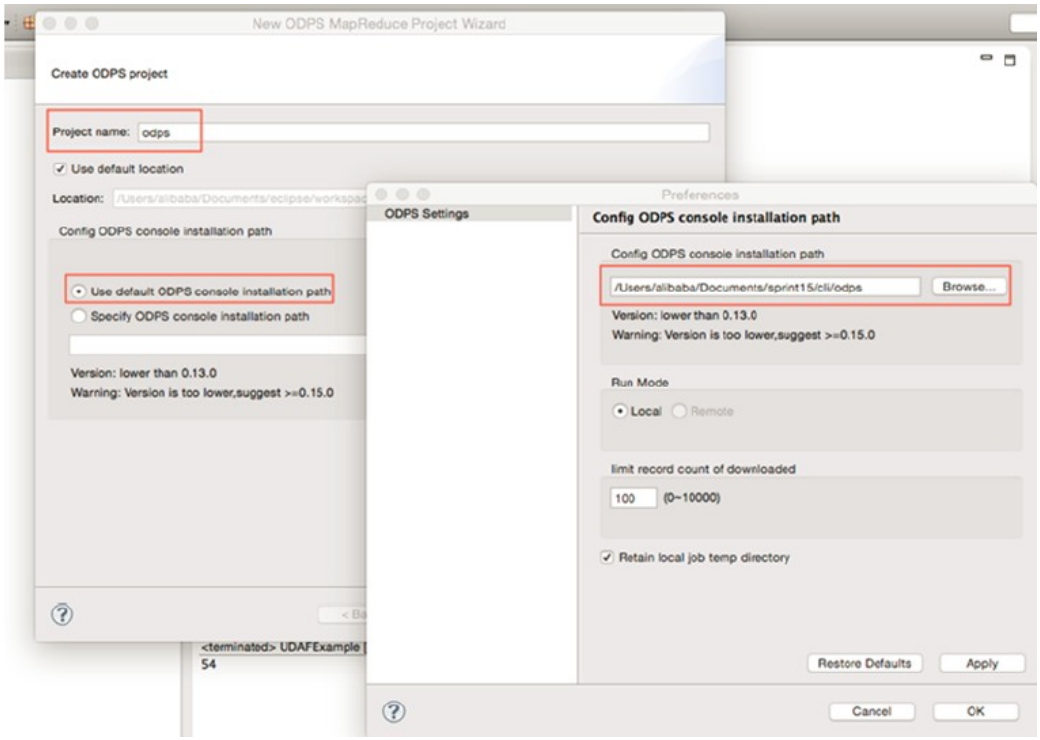
   Step 1

   > **Note** In this example, the project name is ODPS.

2. After the ODPS project is created, a dialog box is displayed, as shown in the following figure. Set Project name, select the path of the MaxCompute client, and then click **Finish**.

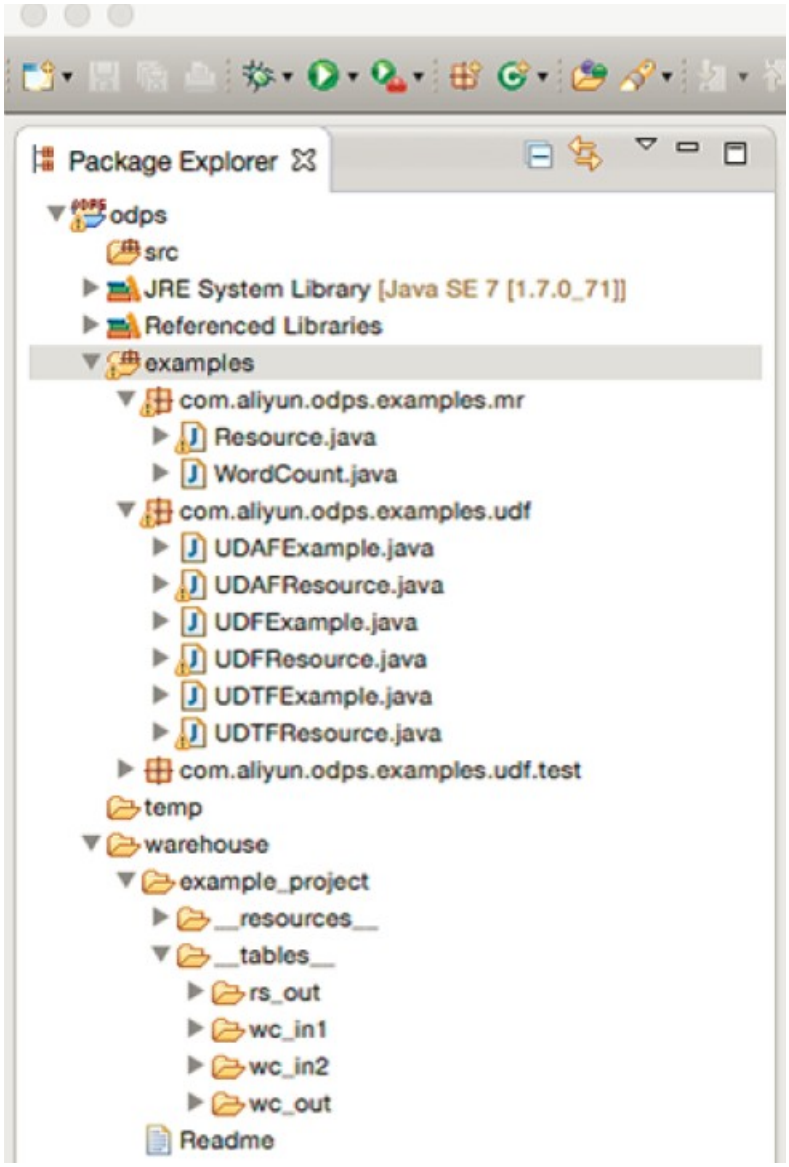
   Step 2



> **Note** The client must be installed in advance.

3. After creating a project, you can see the directory structure on the left-side Package Explorer pane, as shown in the following figure.

   Step 3



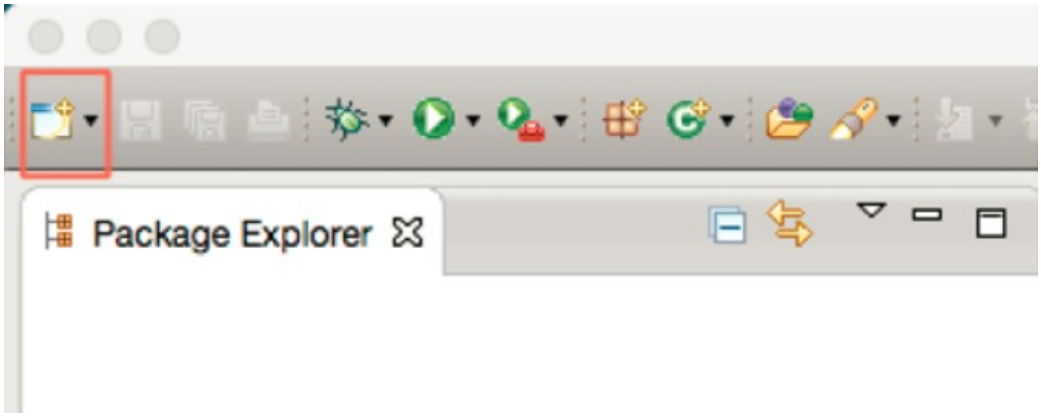
1.21.2.2.2. Method 2

This topic describes the second method to create a project in the Eclipse development plugin.

Procedure

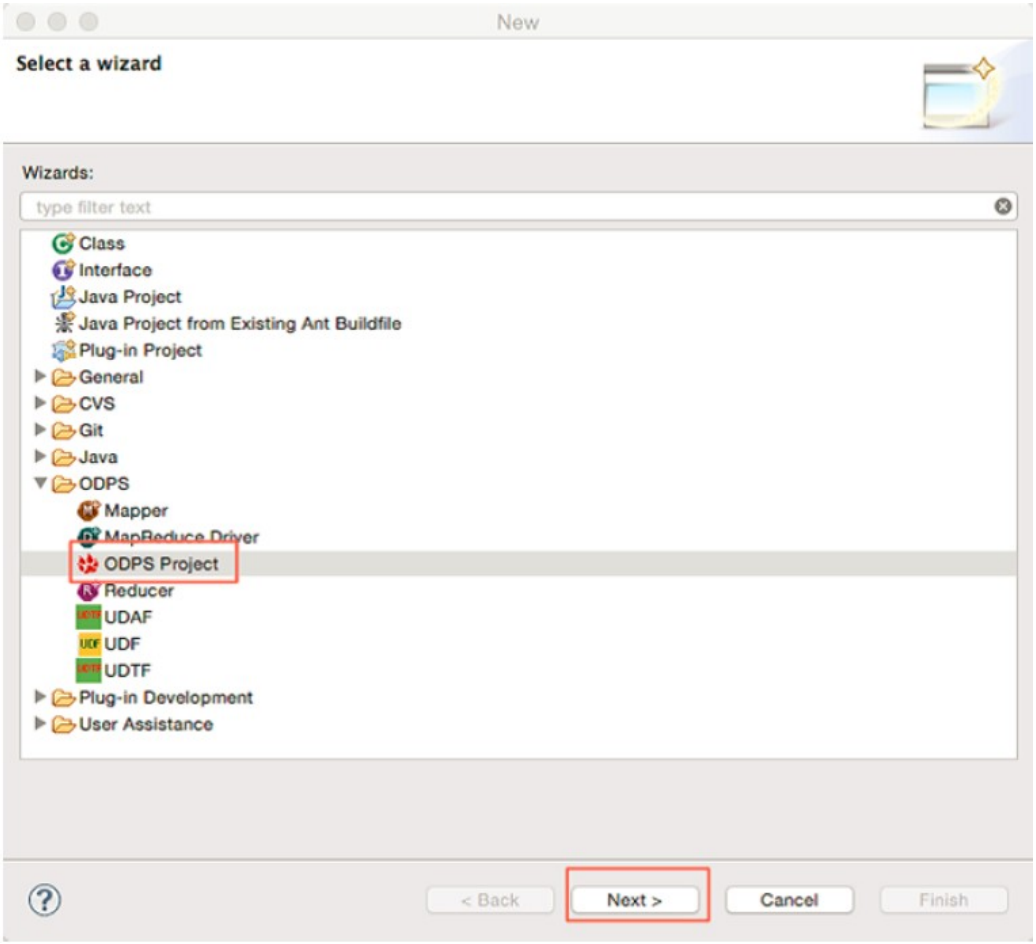
1. Start Eclipse. Click **New** in the upper-left corner, as shown in the following figure.

   Step 1



2. In the dialog box that appears, select **ODPS Project** and click **Next**, as shown in the following figure.

   Step 2



> **Note** In this example, the project name is ODPS.

3. The subsequent steps are the same as those in method 1. After installing the Eclipse plugin, you can use it to compile MapReduce or UDF programs.

> **Note** For a MapReduce running example, see MapReduce running example. For a UDF development and running example, see UDF development and running example.

1.21.2.3. MapReduce running example

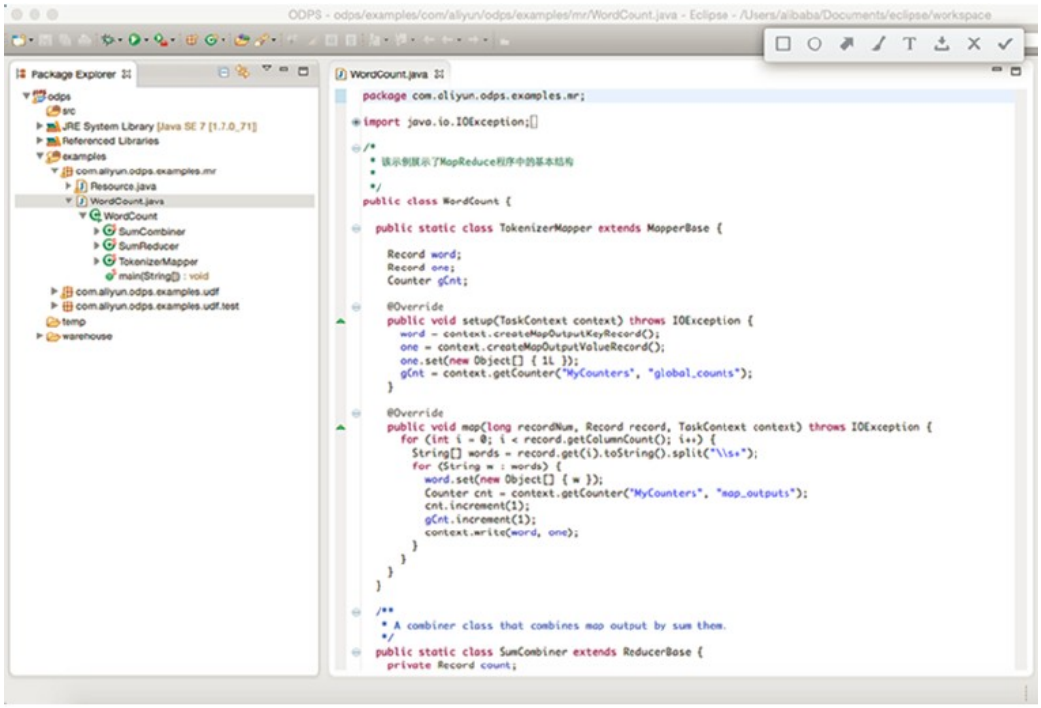
1.21.2.3.1. Quickly run a WordCount example

This topic describes how to use MapReduce to quickly run a WordCount example in the Eclipse plugin.

Procedure

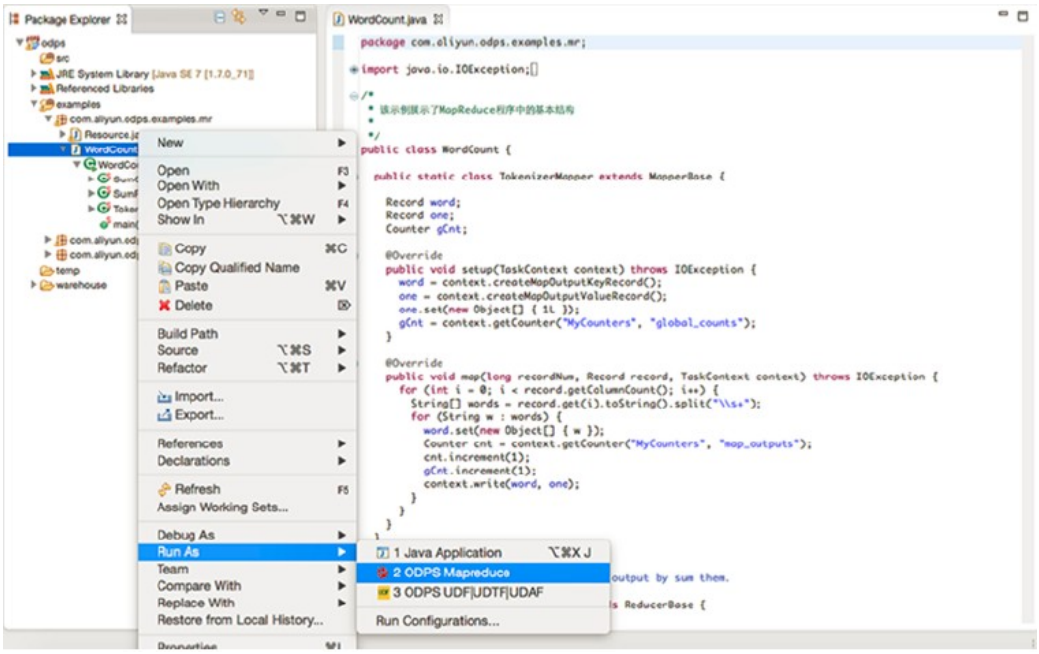
1. Select a WordCount example in MaxCompute, as shown in the following figure.

   WordCount example

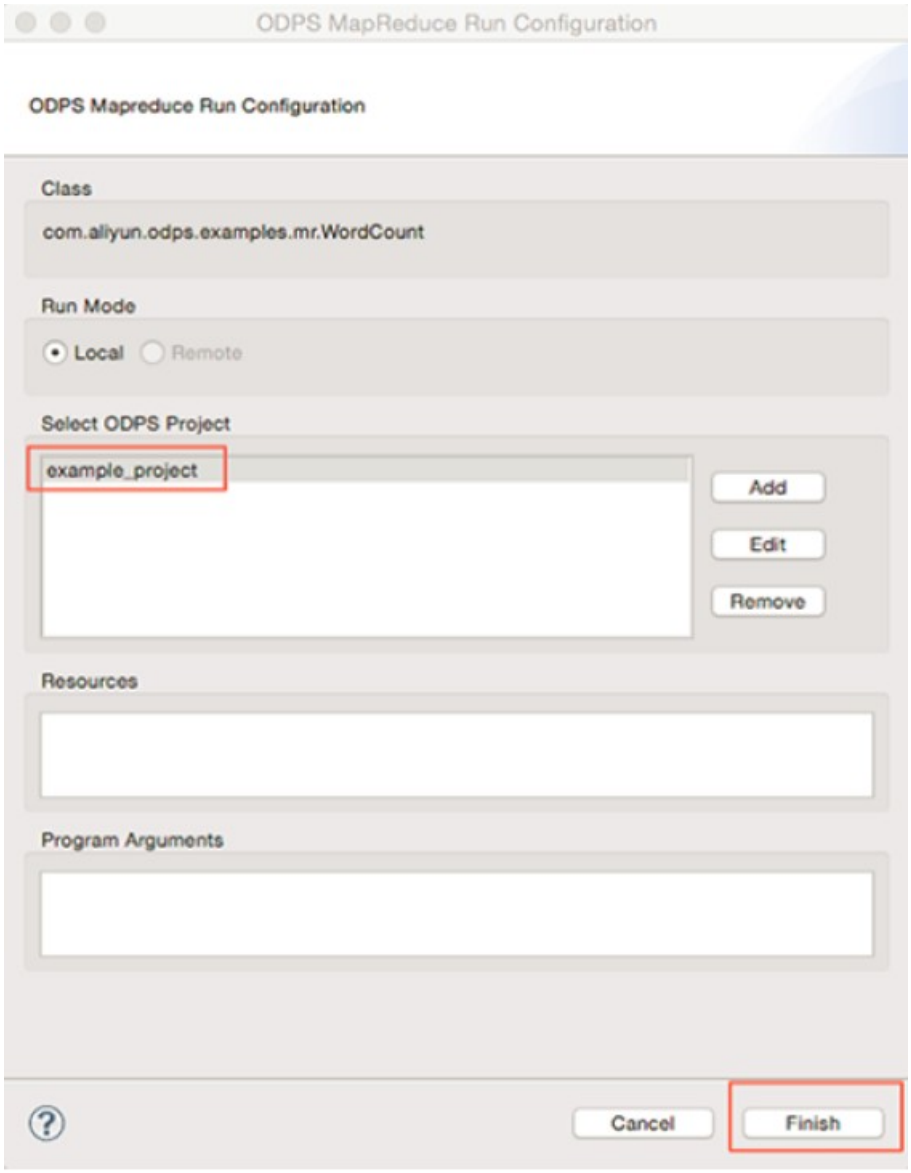
2. Right-click **WordCount.java** and choose **Run AsODPS MapReduce** from the shortcut menu, as shown in the following figure.

   Run the WordCount example



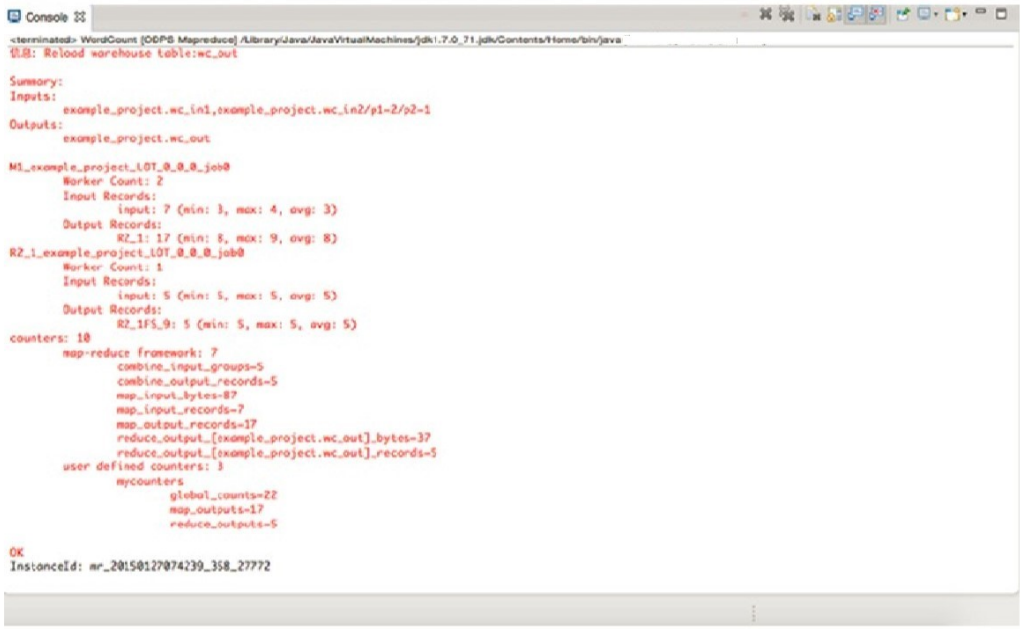
3. In the dialog box that appears, select **example_project** and click **OK**, as shown in the following figure.

   Run the WordCount example



4. The results of the operation are displayed after the operation is executed, as shown in the following figure.

   Execution result of the WordCount example



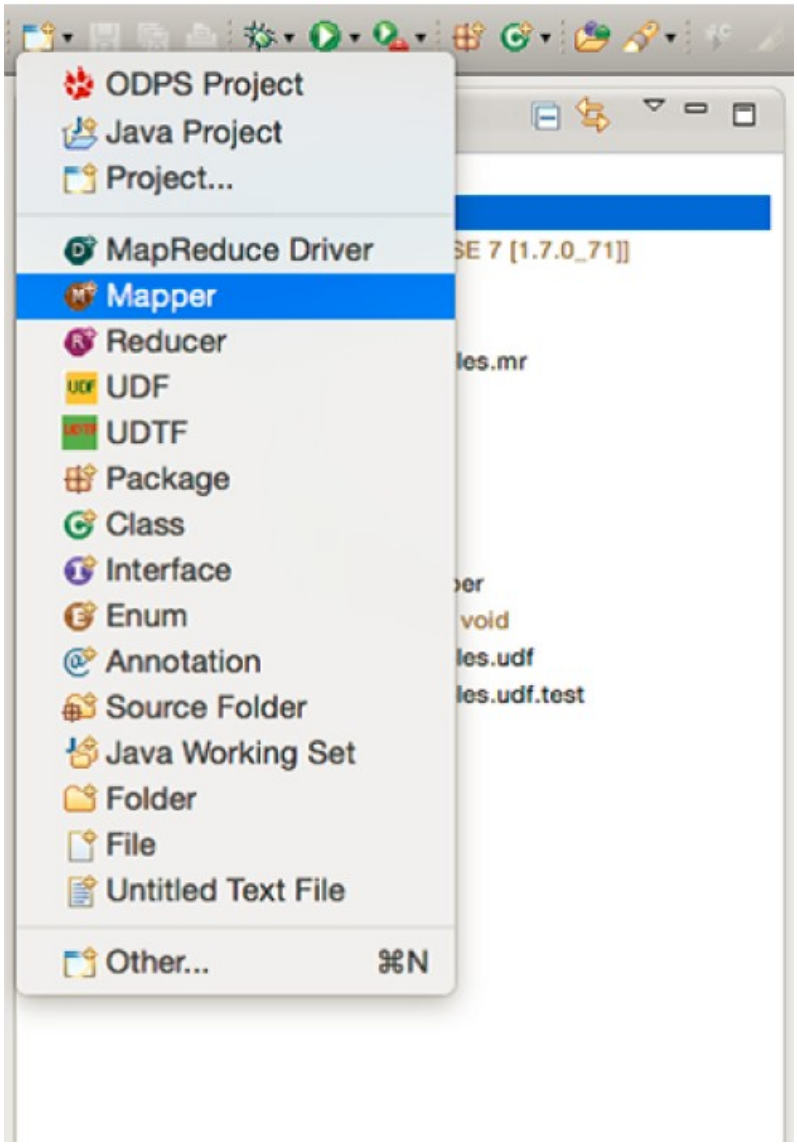
1.21.2.3.2. Run a custom MapReduce program

This topic provides an example on how to run a custom MapReduce program in the Eclipse plugin.

Procedure

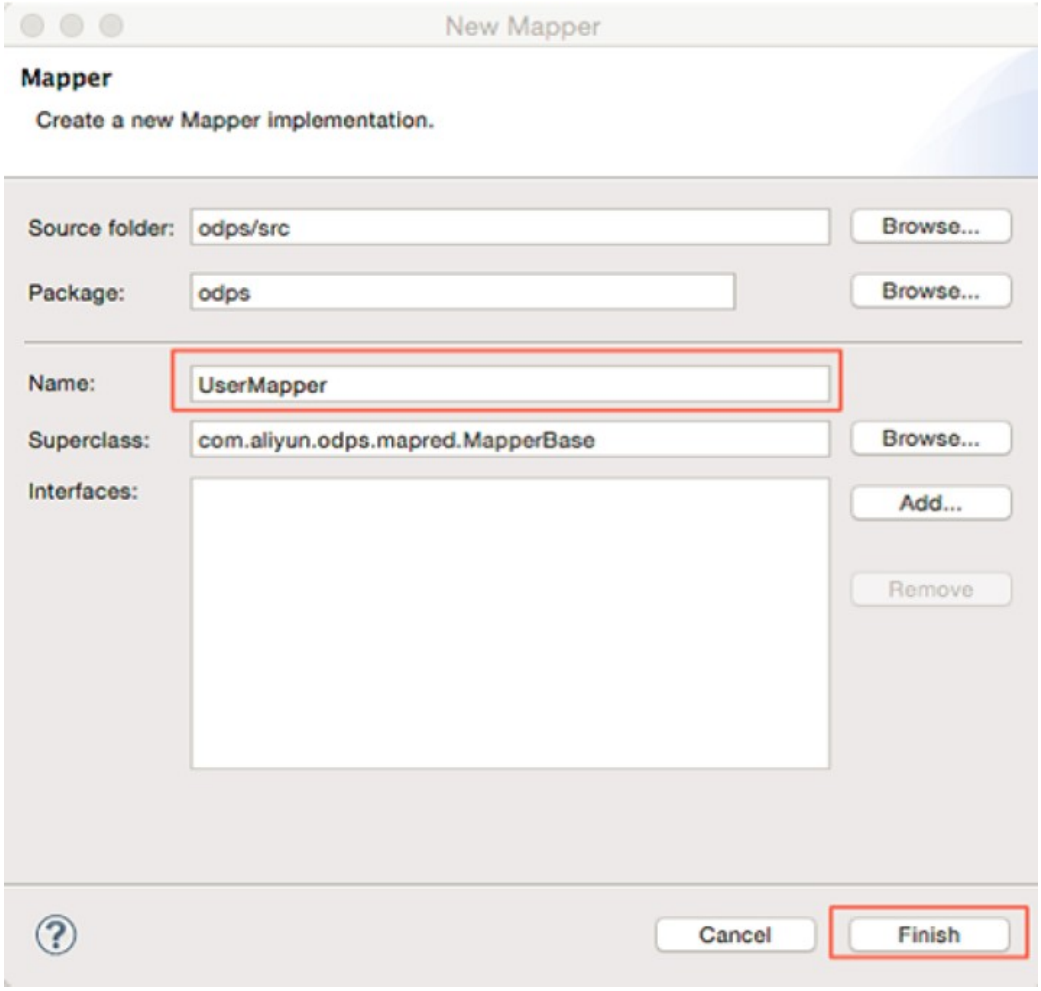
1. In Eclipse, right-click the **src** directory, and choose **New** > **Mapper** from the shortcut menu, as shown in the following figure.

   Step 1



2. Select Mapper. A dialog box is displayed, as shown in the following figure. Enter the name of the Mapper class, and click Finish.

   Step 2

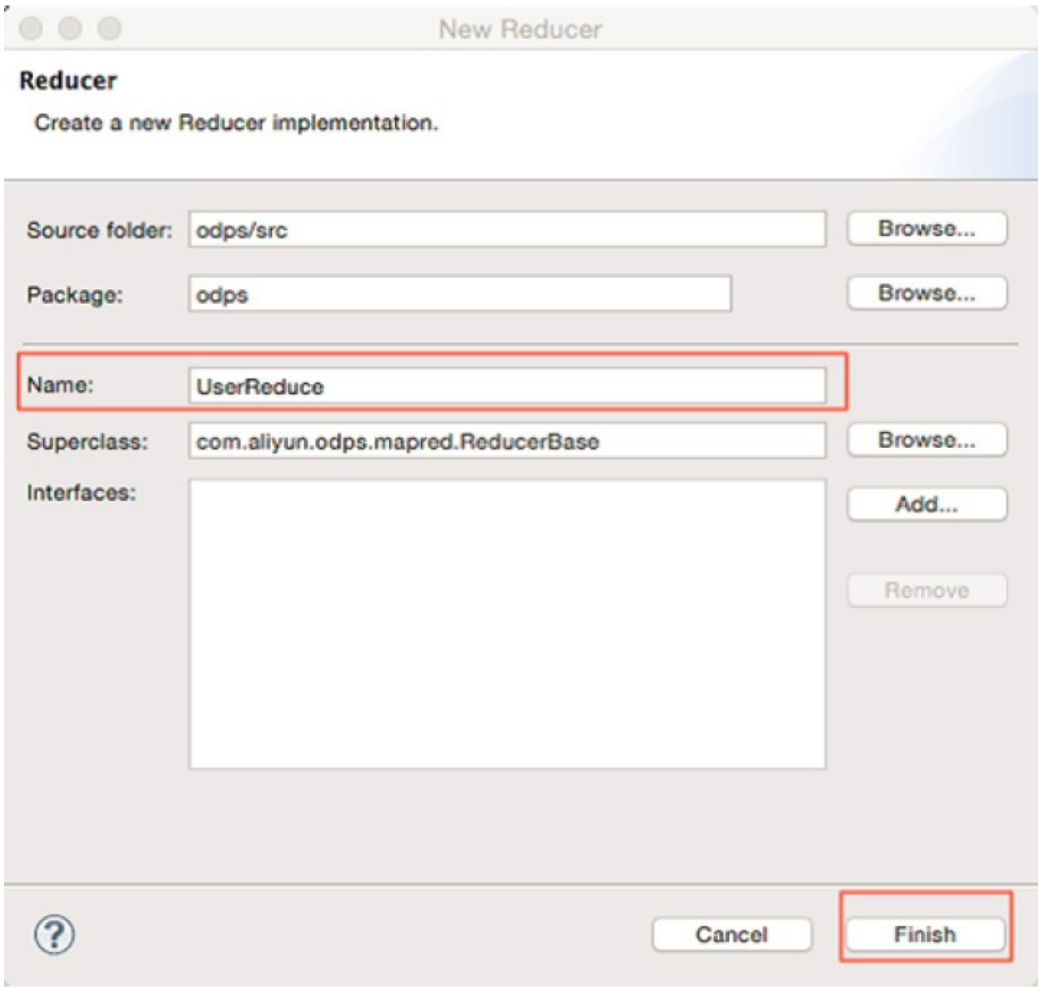


3. On the left-side Package Explorer, the UserReducer.java file is generated in the src directory. The file contains a Mapper class template. The package name is odps by default. The template content is as follows.

```
package odps;
import java.io.IOException;
import com.aliyun.odps.counter.Counter; import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.MapperBase;
public class UserMapper extends MapperBase {
Record word; Record one; Counter gCnt;
@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputKeyRecord(); one = context.createMapOutputValueRecord(); one
.set(new Object[] { 1L });
gCnt = context.getCounter("MyCounters", "global_counts");
}
@Override
public void map(long recordNum, Record record, TaskContext context) throws IOException {
for (int i = 0; i < record.getColumnCount(); i++) { String[] words = record.get(i).toString().split("\\s
+"); for (String w : words) {
word.set(new Object[] { w });
Counter cnt = context.getCounter("MyCounters", "map_outputs"); cnt.increment(1);
gCnt.increment(1); context.write(word, one);
}
}
}
@Override
public void cleanup(TaskContext context) throws IOException {
}
}
```

4. In Eclipse, right-click the **src** directory, and choose **New** > **Reduce** from the shortcut menu, as shown in the following figure.

   Reducer



5. In the dialog box that appears, enter the name of the Reduce class and click Finish.

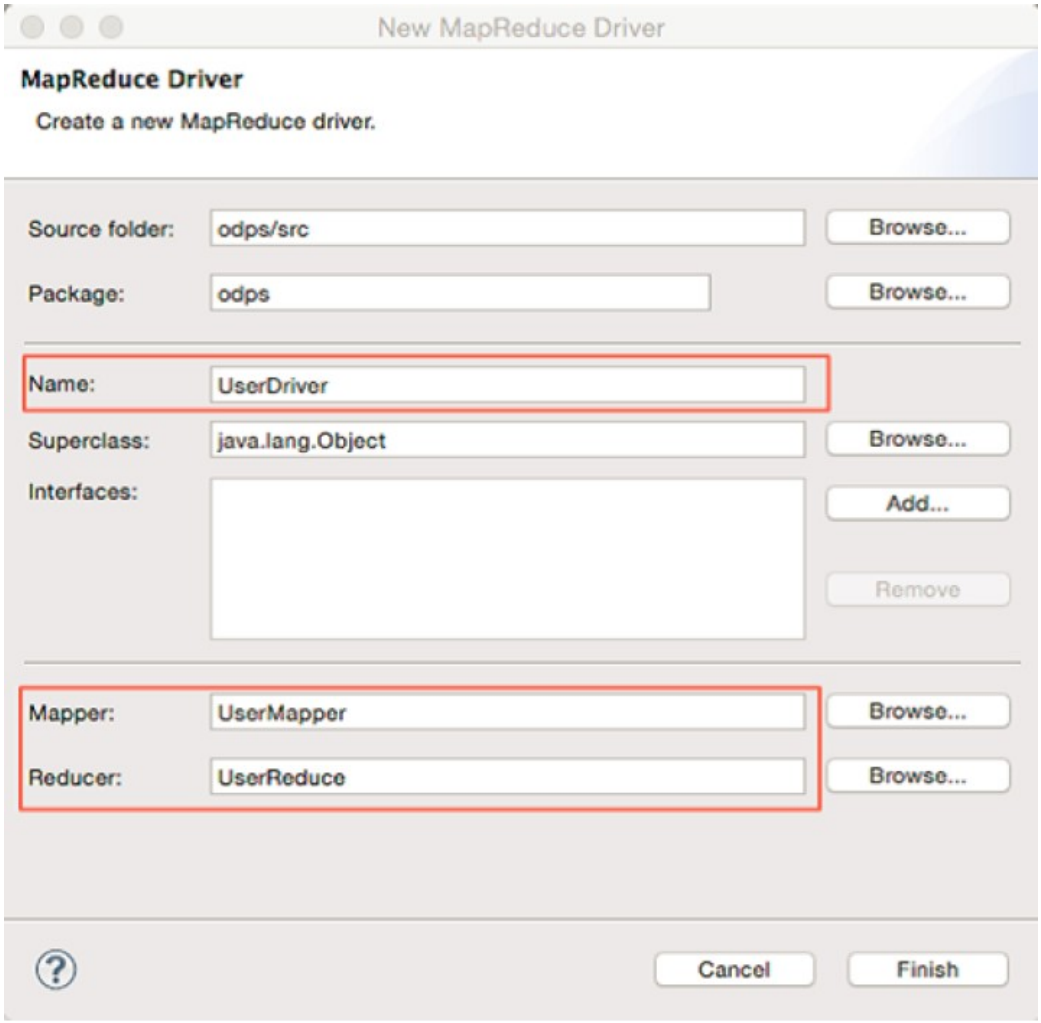
   > **Note** This example uses UserReducer.

6. On the left-side Package Explorer, the UserReducer.java file is generated in the src directory. The file contains a Reduce class template. The package name is odps by default. The template content is as follows:

```
package odps;
import java.io.IOException;
import java.util.Iterator;
import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.ReducerBase;
public class UserReducer extends ReducerBase {
private Record result; Counter gCnt;
@Override
public void setup(TaskContext context) throws IOException { result = context.createOutputRecord
();
gCnt = context.getCounter("MyCounters", "global_counts");
}
@Override
public void reduce(Record key, Iterator<Record> values, TaskContext context) throws IOException
{
long count = 0;
while (values.hasNext()) { Record val = values.next(); count += (Long) val.get(0);
}
result.set(0, key.get(0)); result.set(1, count);
Counter cnt = context.getCounter("MyCounters", "reduce_outputs"); cnt.increment(1);
gCnt.increment(1);
context.write(result);
}
@Override
public void cleanup(TaskContext context) throws IOException {
}
}
```

7. In Eclipse, right-click the **src** directory, and choose **New** > **MapReduce Driver** from the shortcut menu.
8. A dialog box is displayed, as shown in the following figure. Set Name, Mapper, and Reducer, and then click Finish.

   MapReduce Driver

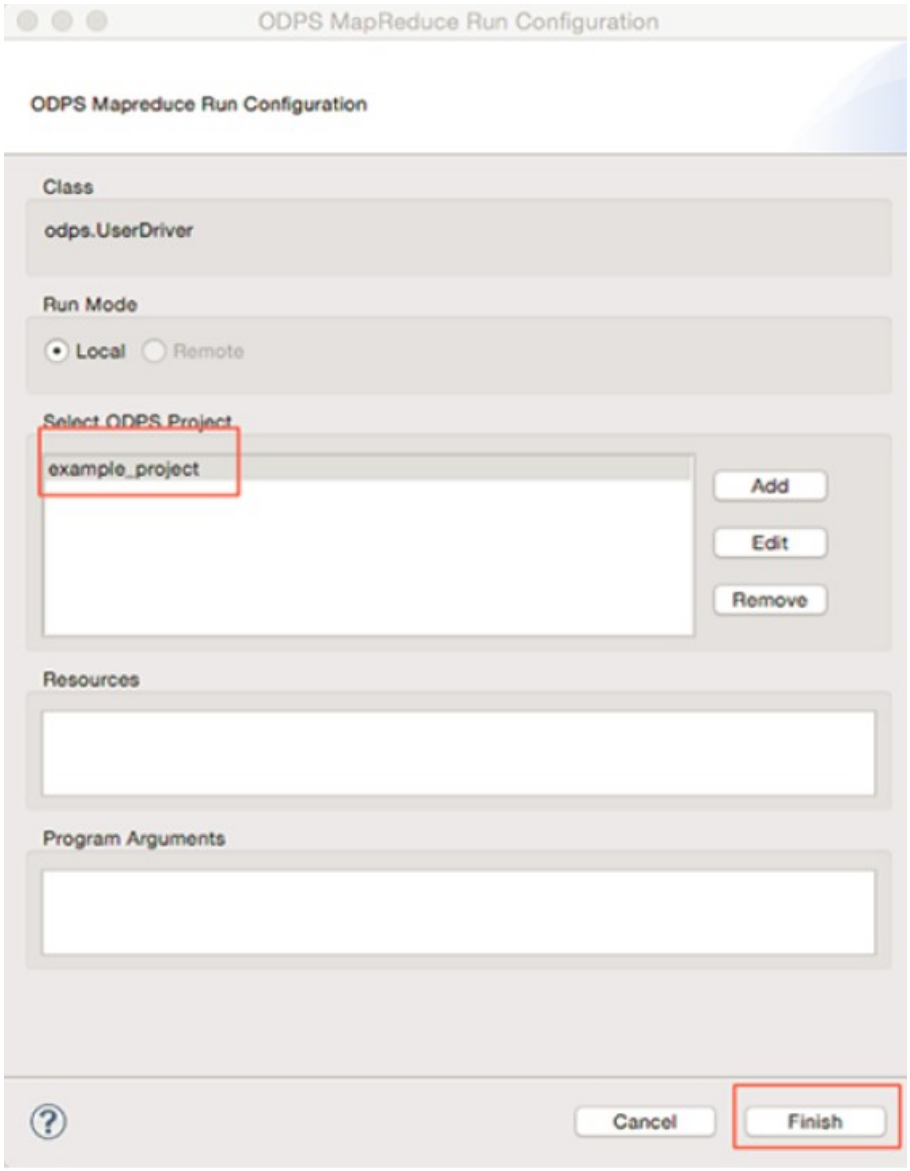


9. On the left-side Package Explorer, the MyDriver.java file is generated in the src directory. The file contains a MapReduce Driver template. The package name is odps by default. The template content is as follows:

```
package odps;
import com.aliyun.odps.OdpsException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.examples.mr.WordCount.SumCombiner;
import com.aliyun.odps.examples.mr.WordCount.SumReducer;
import com.aliyun.odps.examples.mr.WordCount.TokenizerMapper;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.RunningJob;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;
public class UserDriver {
public static void main(String[] args) throws OdpsException { JobConf job = new JobConf();
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(SumCombiner.class);
job.setReducerClass(SumReducer.class);
job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"));
job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigint"));
InputUtils.addTable(
TableInfo.builder().tableName("wc_in1").cols(new String[] { "col2", "col3" }).build(), job);
InputUtils.addTable(TableInfo.builder().tableName("wc_in2").partSpec("p1=2/p2=1").build(), job);
OutputUtils.addTable(TableInfo.builder().tableName("wc_out").build(), job);
RunningJob rj = JobClient.runJob(job); rj.waitForCompletion();
}
}
```

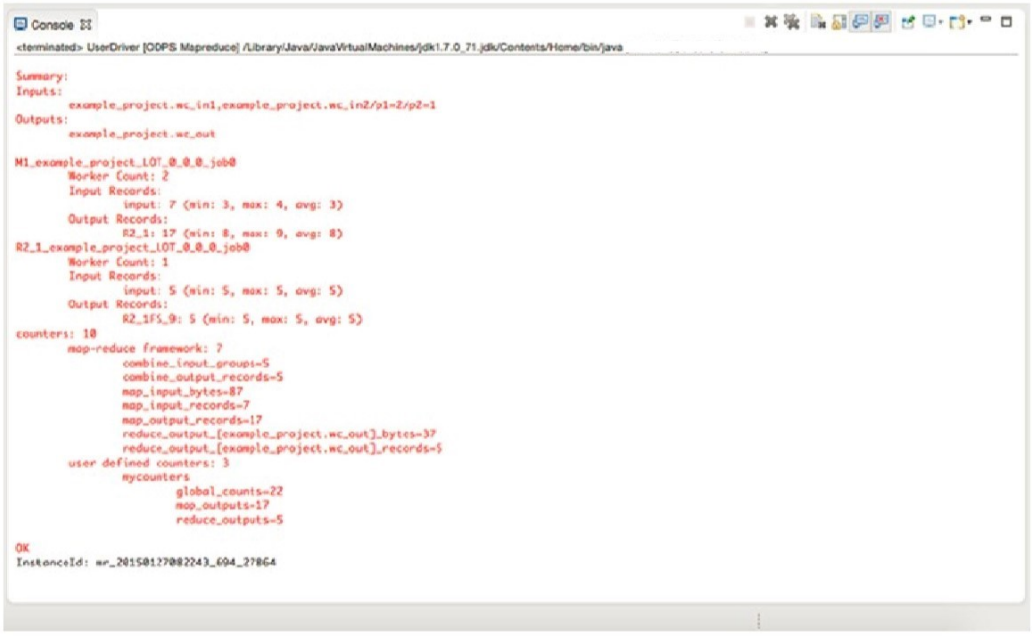
10. Run the MapReduce program. Right-click UserDriver.java and choose **Run As > ODPS MapReduce** from the shortcut menu. In the dialog box that appears, click **OK**. A dialog box is displayed, as shown in the following figure.

    ODPS MapReduce Run Configuration



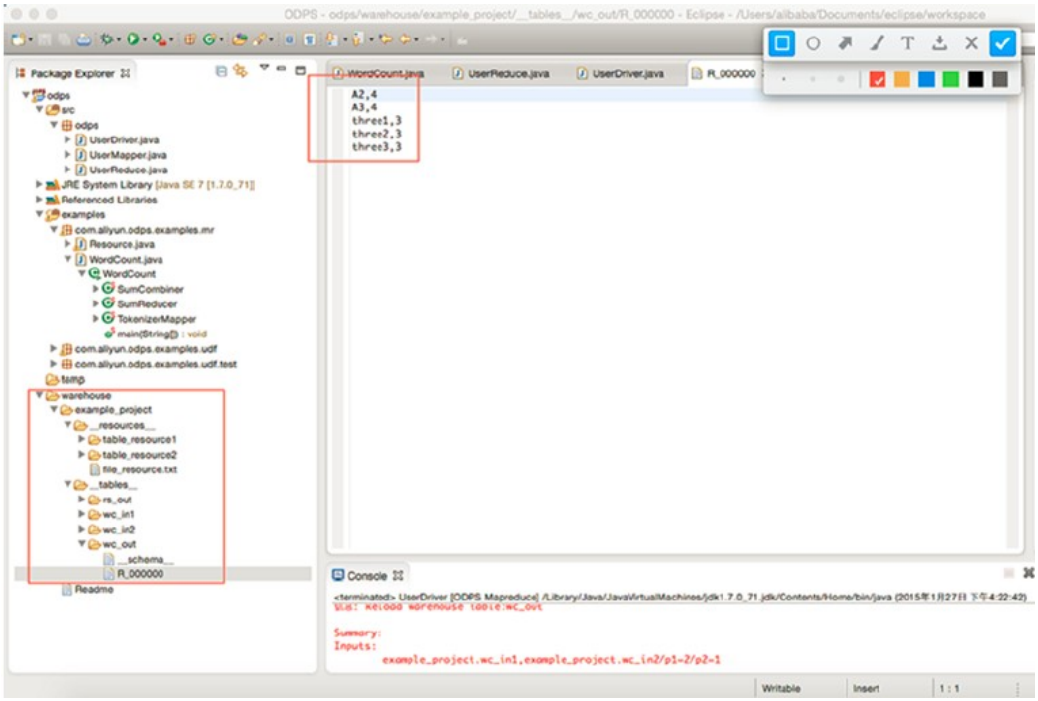
11. Select example_project as the MaxCompute project. Click **Finish** to start running the MapReduce program locally. If the output is as shown in the following figure, the local running is successful.

    Console



12. The execution result is stored in the warehouse directory. Refresh the ODPS project, as shown in the following figure.

    Output

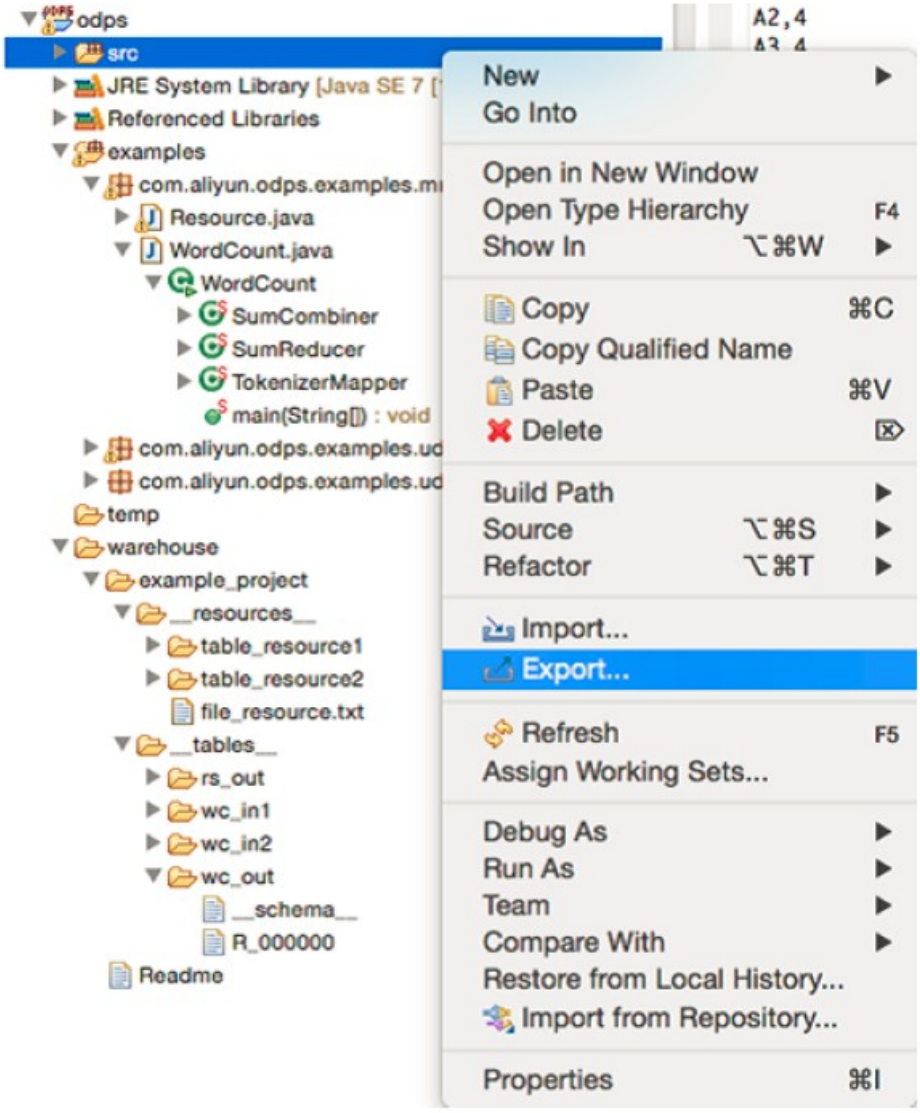


> **Note** wc_out is the output directory, and R_000000 is the result file. After confirming that the result is correct through local debugging, you can use the export function of Eclipse to package the MapReduce program for subsequent use in the distributed environment.

13. The export procedure is as follows:

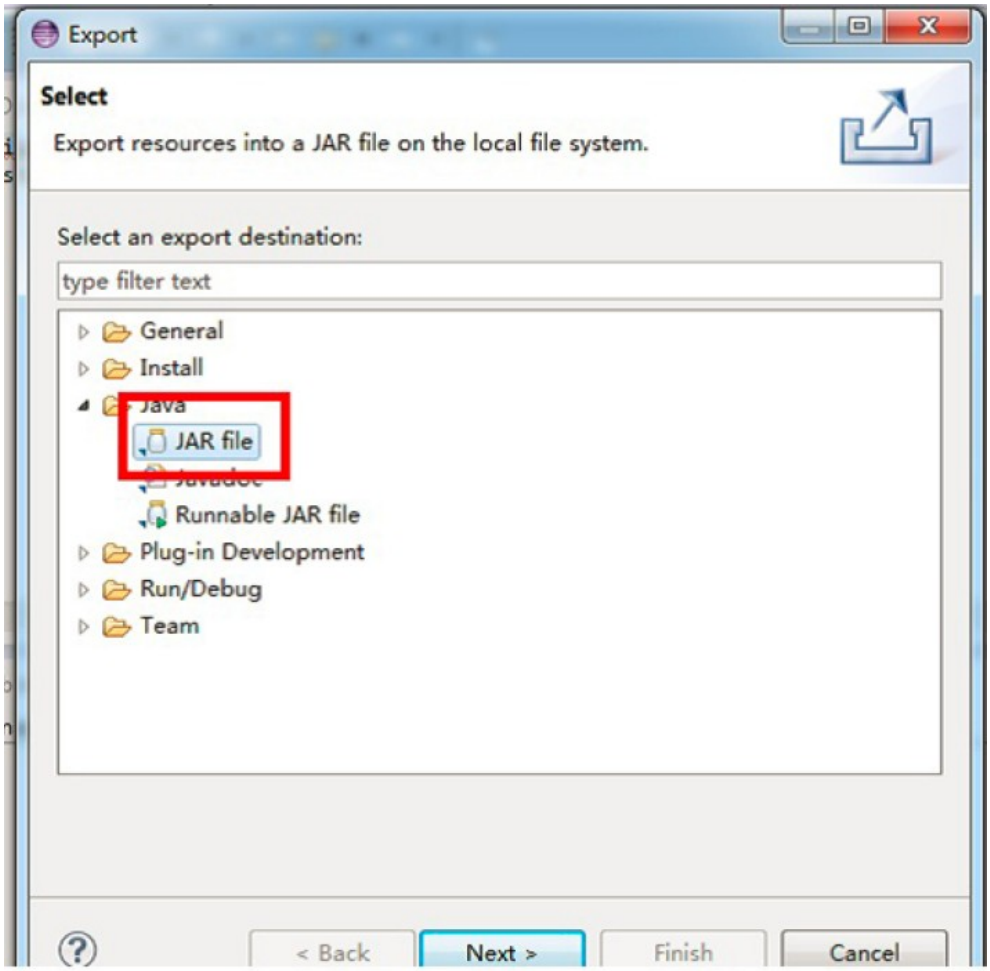
i. Right-click the src directory and select **Export** from the shortcut menu, as shown in the following figure.

Step 1



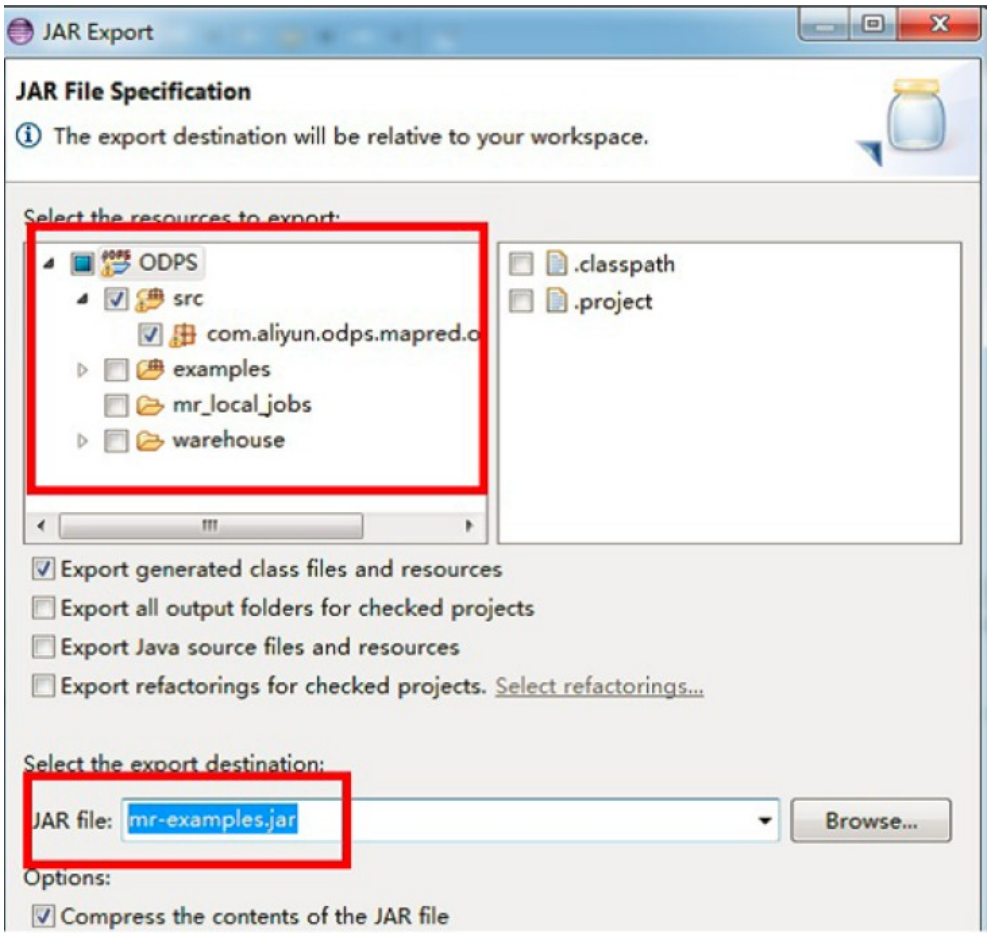
ii. Select JAR file as the export destination, as shown in the following figure.

Step 2



iii. You only need to export the package (com.aliyun.odps.mapred.open.example) in the src directory. Set JAR file to mr- examples.jar, as shown in the following figure.

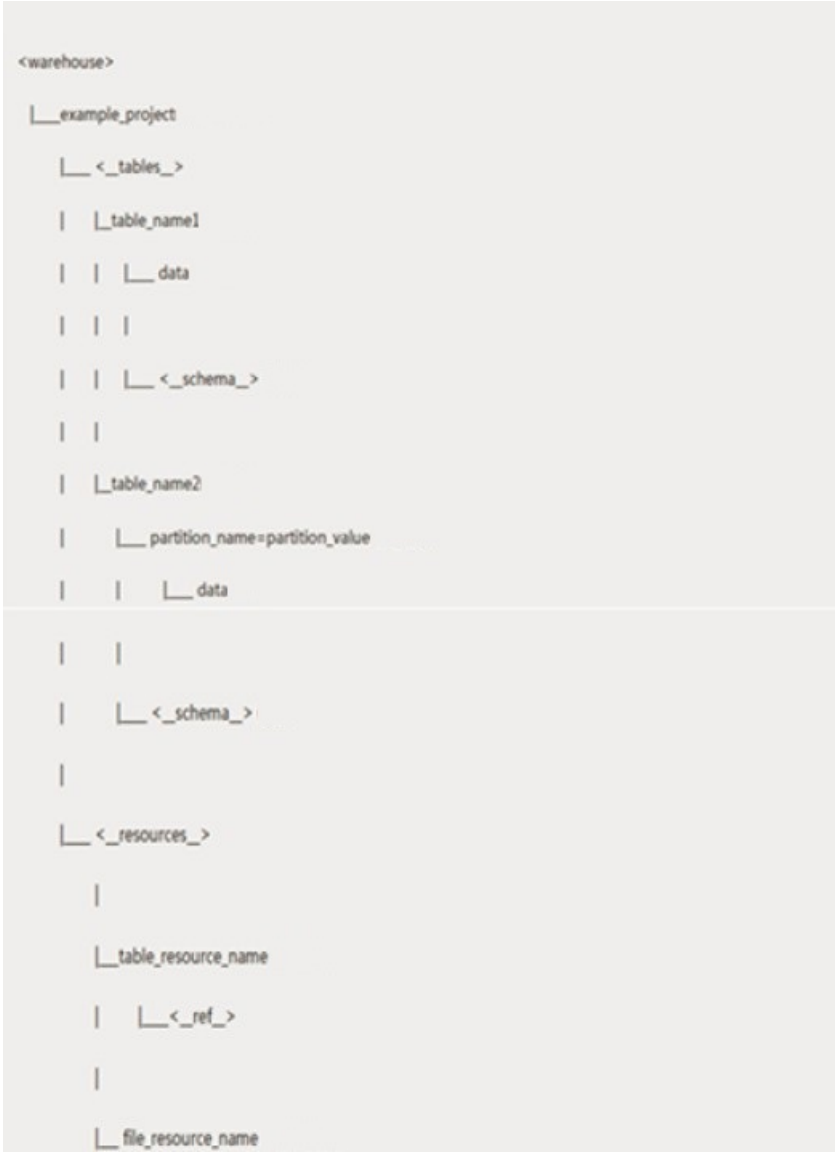
Step 3



> **Note** In this example, the program package is named mr-examples.jar. You can name the package based on your actual requirements.

iv. Click OK. The export process is complete.

14. If you want to create a new project locally, you can create a new subdirectory (at the same level as example_project) under warehouse. The following figure shows the directory structure.



An example of the schema file:

Non-partitioned table:

```
project=project_name table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
-- Partitioned table: project=project_name table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING partitions=col1:BIGIN
T,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
-- Note that the following data formats are supported: bigint, double, boolean, datetime, and stri
ng. These formats correspond to the following Java data types: long, double, boolean, java.util.Da
te, and java.lang.String.
```

An example of the data file:

```
1,1.1,true,2015-06-04 11:22:42 896,hello world
\N,\N,\N,\N,\N
-- Note that the time is in milliseconds. For all time formats, \N is used to represent NULL.
```

> **Note**
>
> - Run the MapReduce program locally. The warehouse directory is checked for data tables or resources by default. If the tables or resources do not exist, data of the server is downloaded to the warehouse directory. Then the program runs locally.
> - After running MapReduce, refresh the warehouse directory to view the generated result.

1.21.2.4. UDF development and running example

1.21.2.4.1. Local debug UDF programs

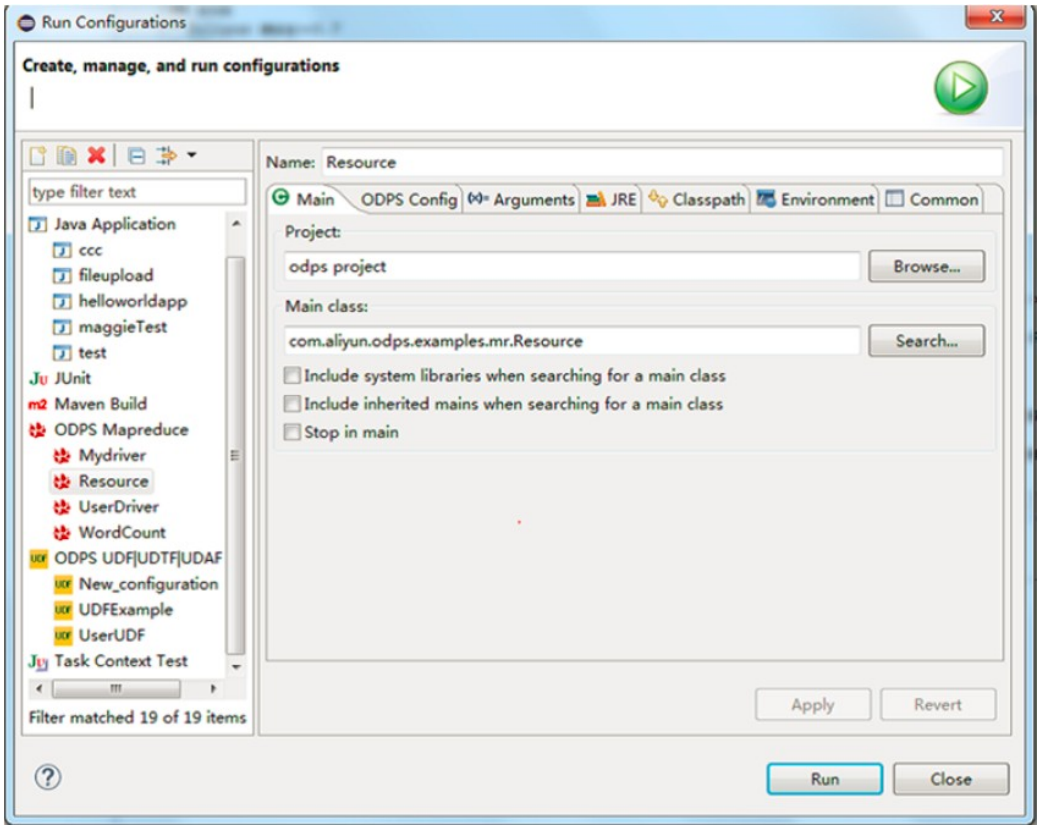
1.21.2.4.1.1. Run a UDF from the menu bar

This topic describes how to quickly run a UDF from the menu bar of the Eclipse plugin.

Procedure

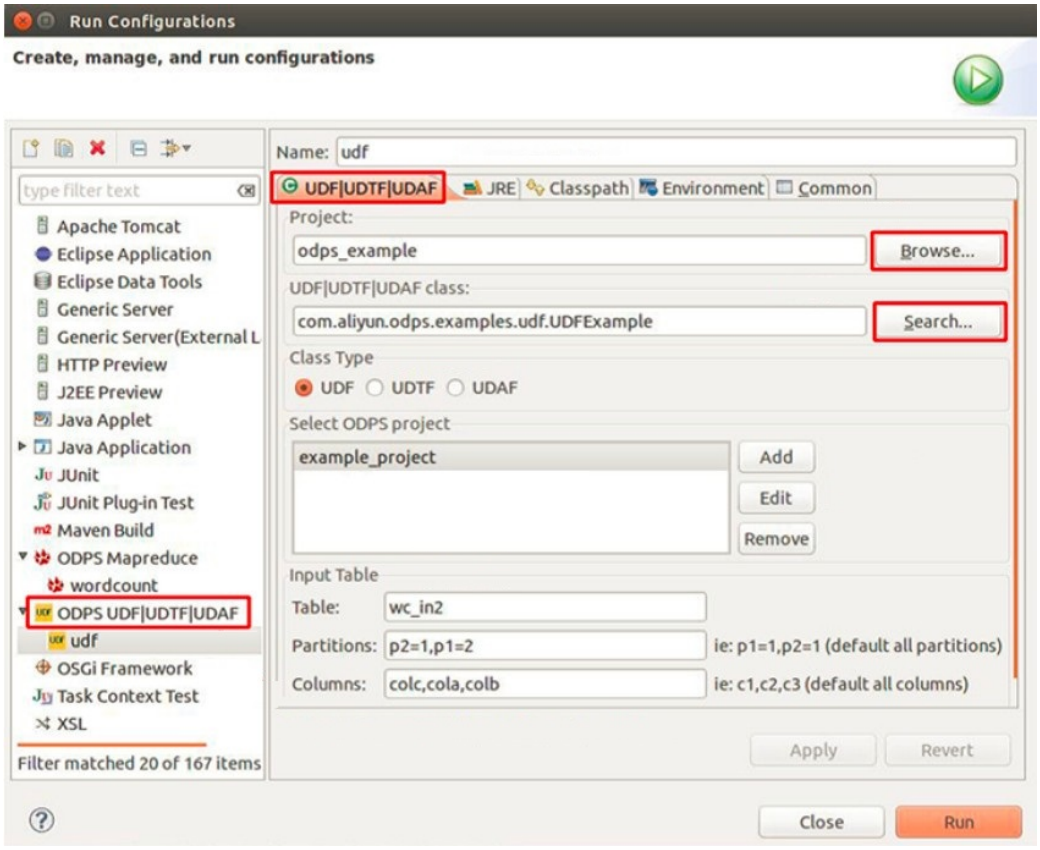
1. Choose **Run** > **Run Configurations** from the menu bar. A dialog box is displayed, as shown in the following figure.

   Run Configurations 1



2. To create a run configuration, select the UDF class and type to be run, select an ODPS project, and fill in the input table information, as shown in the following figure.

   Run Configurations 2



Note There are three parameters in the Input Table area: Enter the input table of the UDF in Table. Enter the partitions from which data is read in Partitions. Separate multiple partitions by commas (,). Enter the columns in which data is transmitted as UDF parameters in Columns. Separate multiple columns by commas (,).

3. Click **Run**. The running result is displayed in the console, as shown in the following figure.

   Console

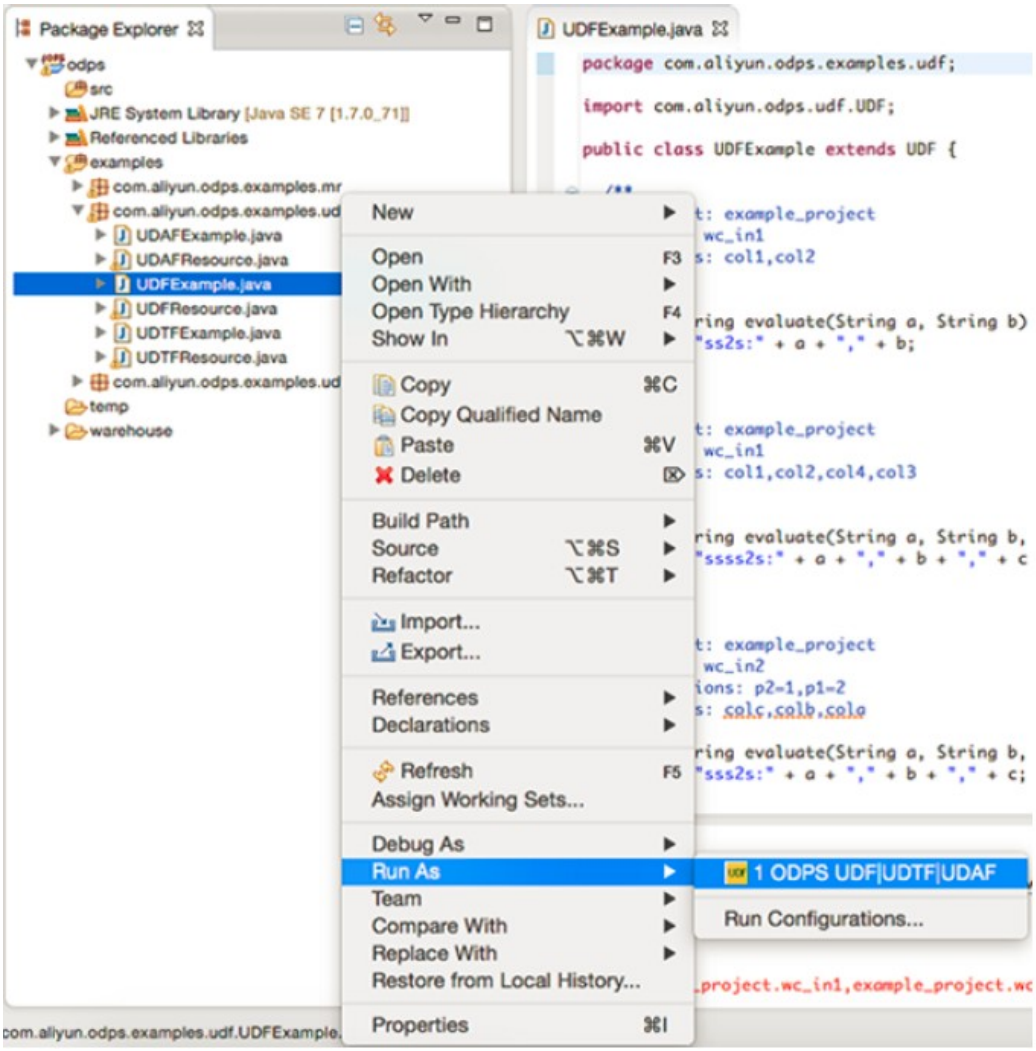
1.21.2.4.1.2. Use the right-click shortcut menu to quickly run a UDF

This topic describes how to use the right-click shortcut menu to quickly run a UDF in the Eclipse development plugin.

Procedure

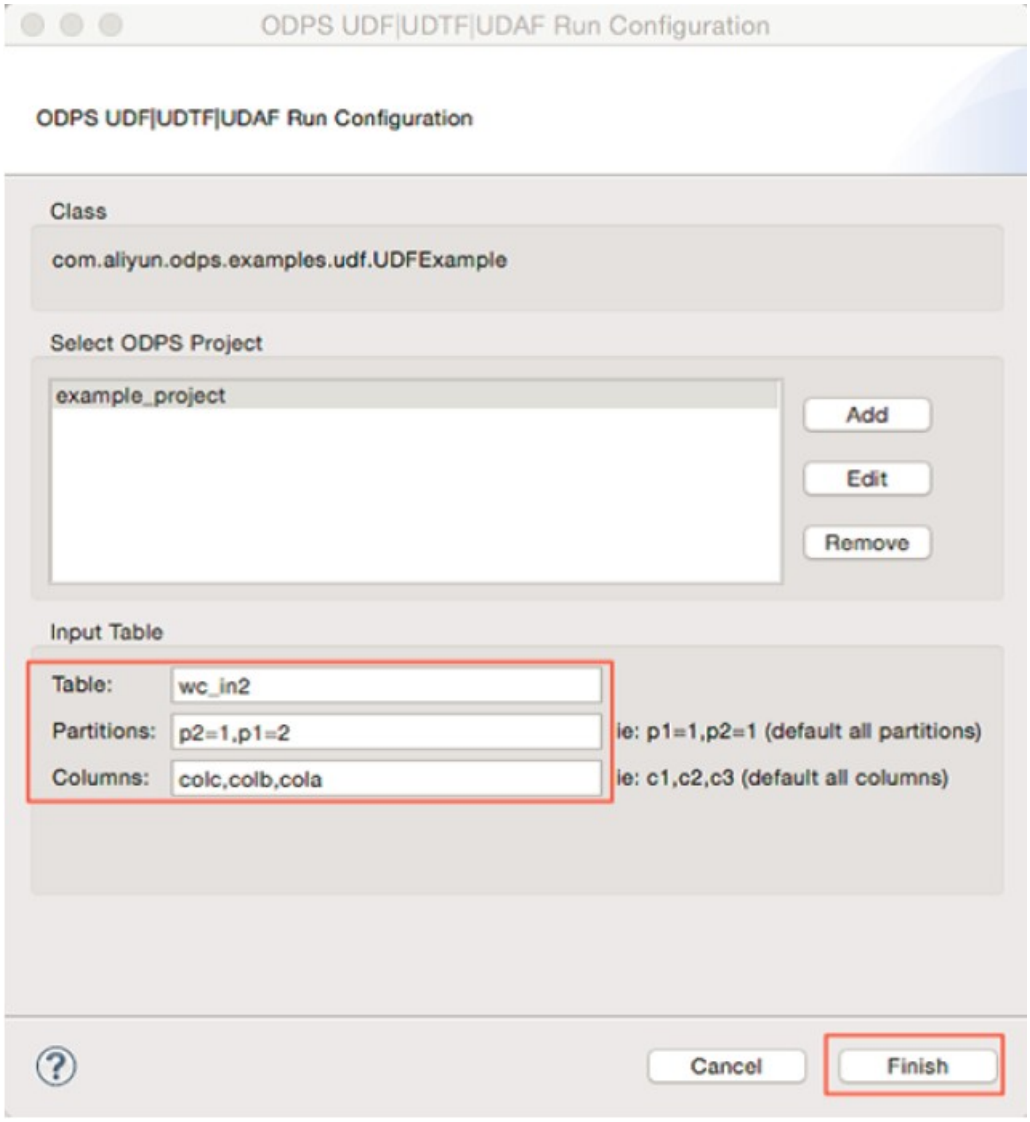
1. Right-click a udf.java file (such as UDFExample.java) and choose **Run As > Run UDF|UDAF|UDTF** from the shortcut menu, as shown in the following figure.

Step 1



2. In the dialog box that appears, configure the relevant parameters, as shown in the following figure.

Step 2



Note Table indicates the input table of the UDF. Partitions indicate the partitions from which data is read. Multiple partitions are separated by commas. Columns indicate the columns from which data is read. Multiple columns are separated by commas. These parameters are imported to the UDF as parameters.

3. Click **Finish** to run the UDF and obtain the result.

1.21.2.4.2. Run a UDF program

This topic describes how to run a UDF program in the Eclipse development plugin.

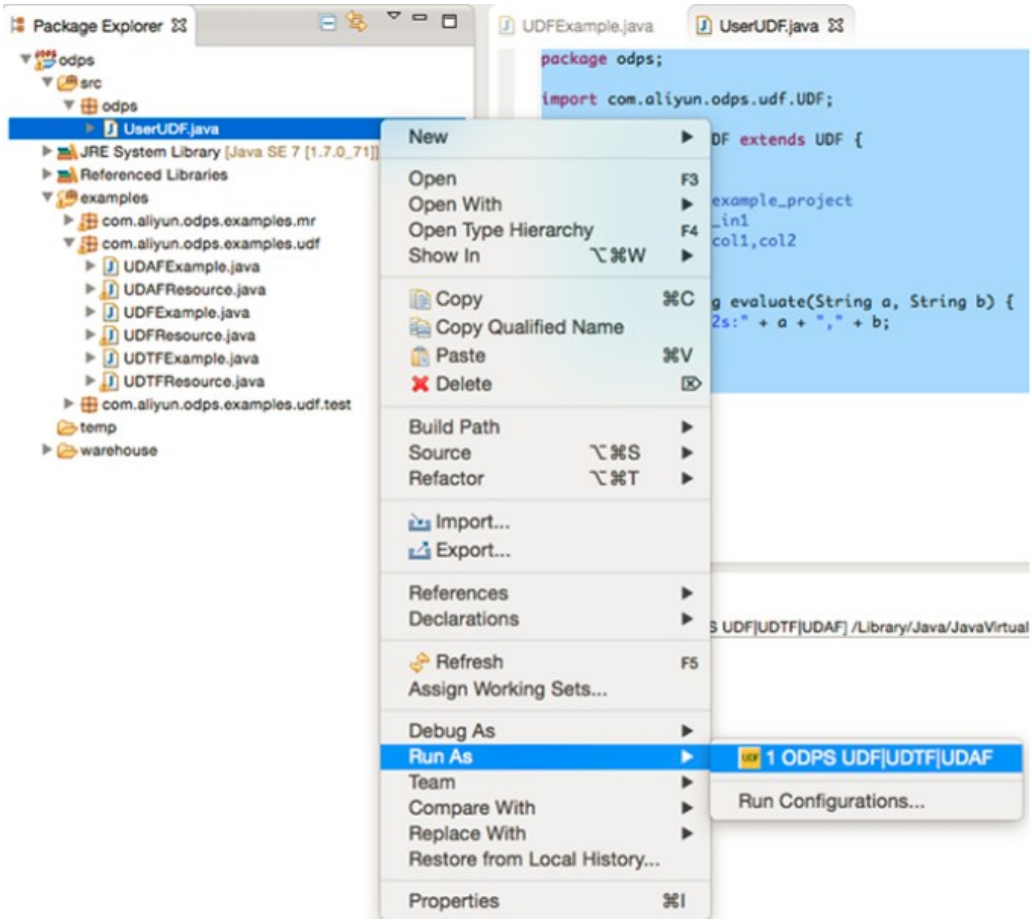
Procedure

1. Right-click a project and choose **New** > **UDF** (or choose **File** > **New** > **UDF** from the menu bar). Enter a UDF class name and click **Finish**. A Java file with the same name as the UDF class is generated in the **src** directory. Edit the content of the java file as follows:

```
package odps;
import com.aliyun.odps.udf.UDF;
public class UserUDF extends UDF {
/**
* project: example_project
* table: wc_in1
* columns: col1,col2
*
*/
public String evaluate(String a, String b) { return "ss2s:" + a + "," + b;
}
}
```

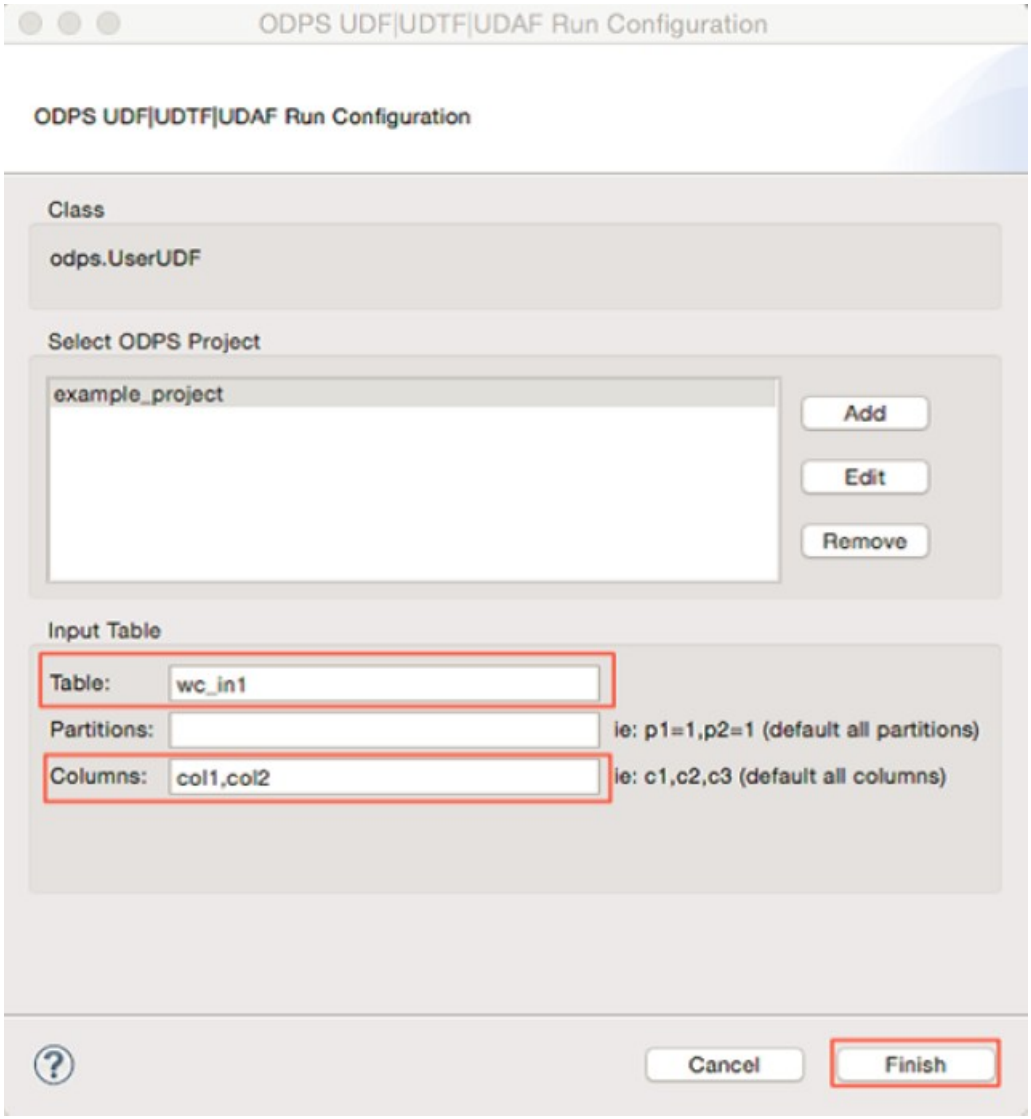
2. Right-click the Java file (such as UserUDF.java) and choose **Run As > ODPS UDF|UDTF|UDAF** from the shortcut menu, as shown in the following figure.

   Step 1



3. In the dialog box that appears, configure the relevant parameters, as shown in the following figure.

   Step 3-1



4. Click **Finish** to obtain the result.

```
ss2s:A1,A2
ss2s:A1,A2
ss2s:A1,A2
ss2s:A1,A2
```

> **Note** This example shows how to run a UDF program. You can use the same method to run a UDTF program.

1.21.2.5. Graph running example

After creating a MaxCompute project, you can write your own Graph program and debug it locally by performing the following steps.

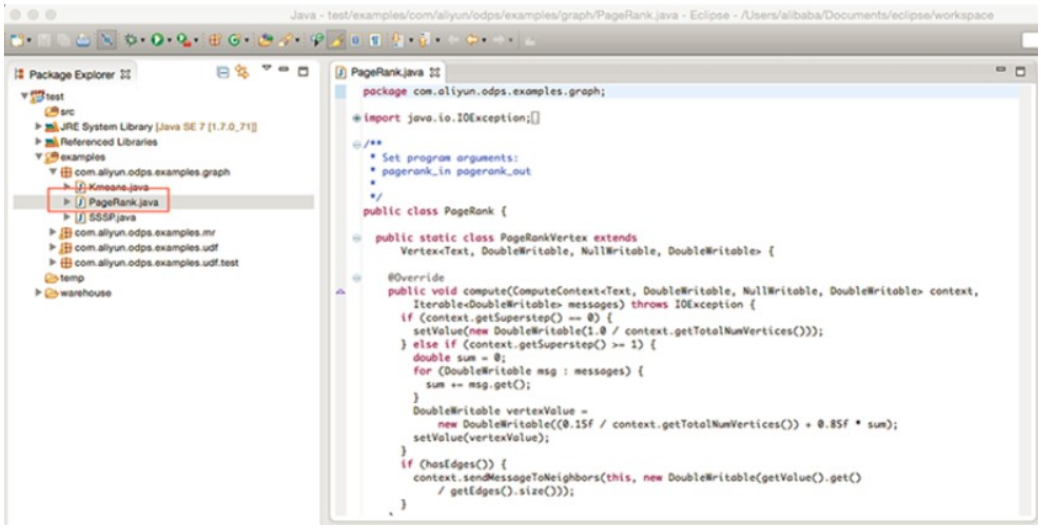
Context

In this example, you can use PageRank.java provided by the plugin to perform local debugging.

Procedure

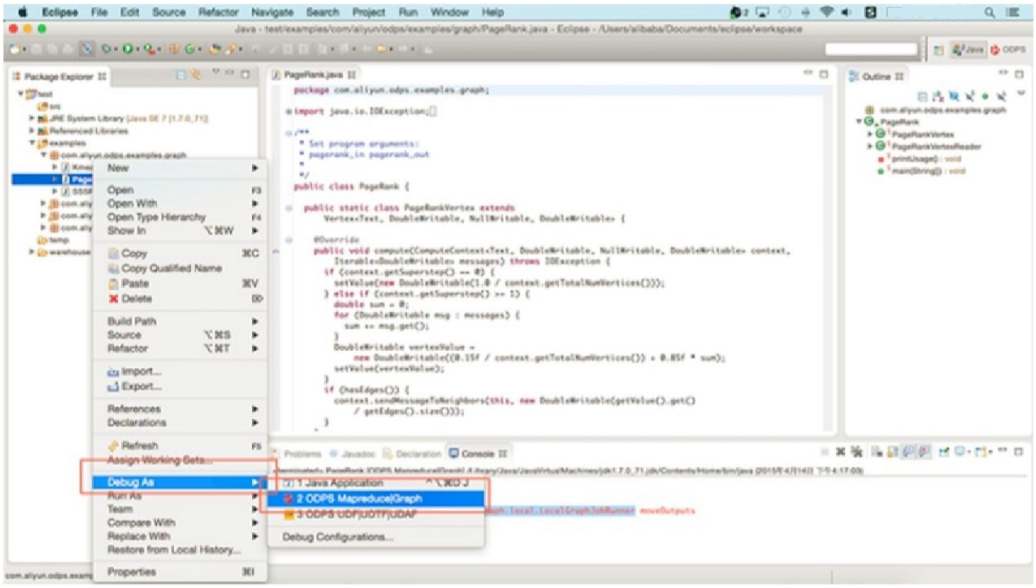
1. Choose **examples > PageRank.java**, as shown in the following figure.

   PageRank.java code 1

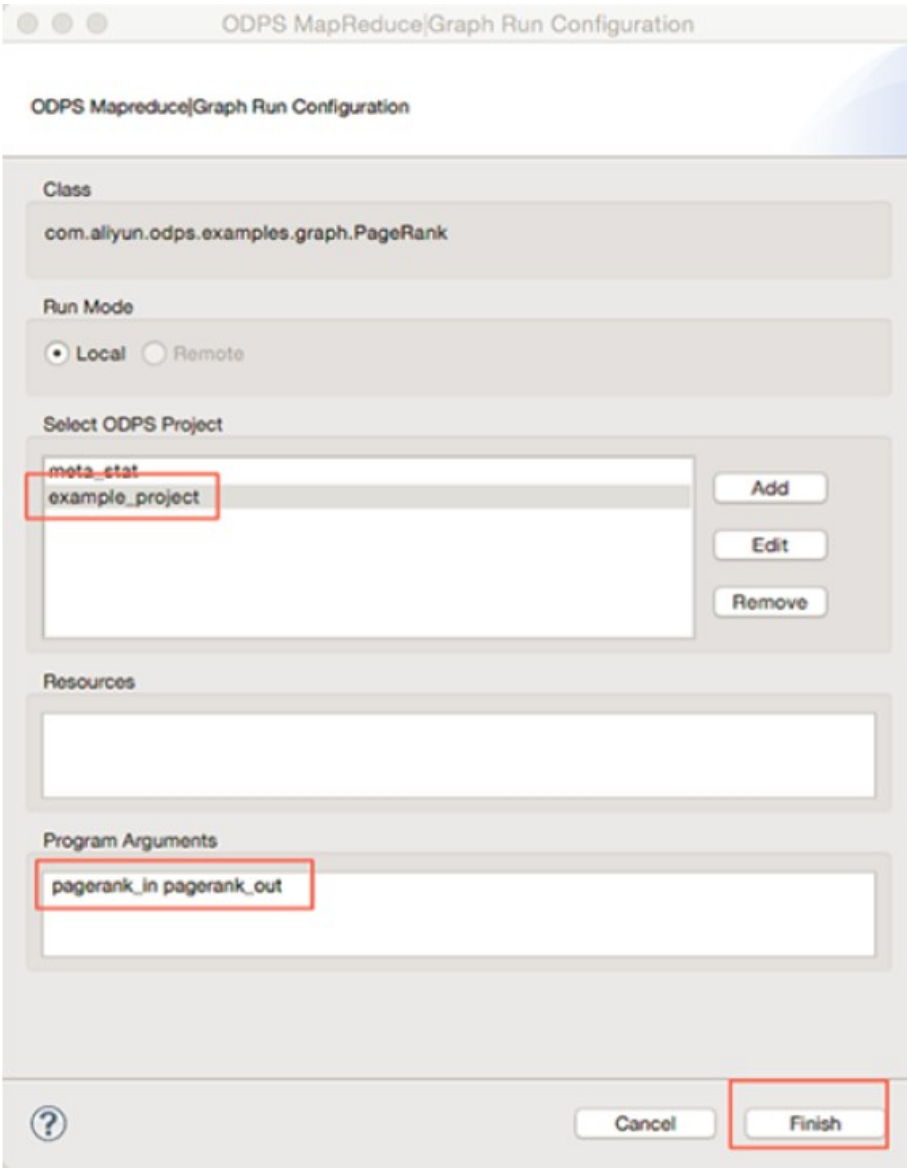
2. Right-click it and choose **Debug As > ODPS MapReduce|Graph**, as shown in the following figure.

   PageRank.java code 2

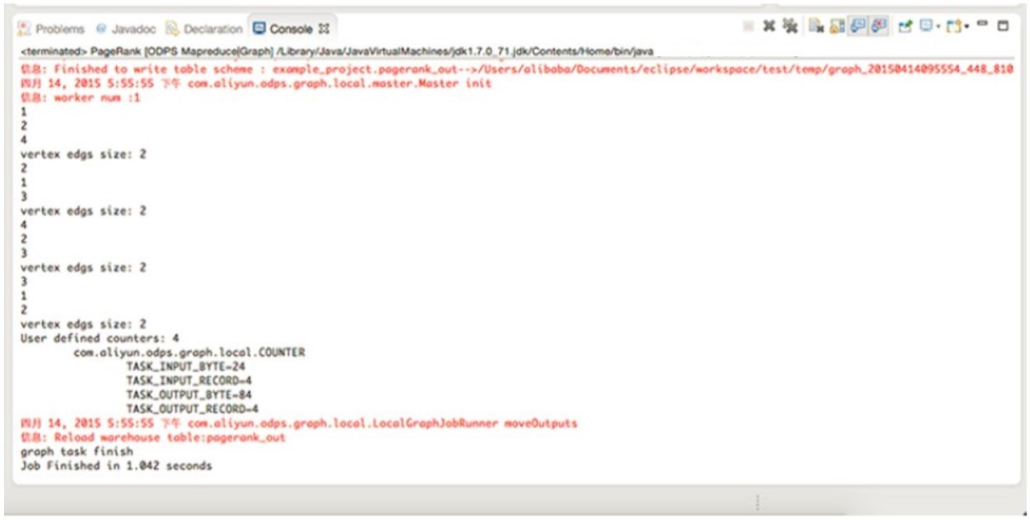
3. In the pop-up dialog box, enter the information as shown below.

   Configuration Drawings



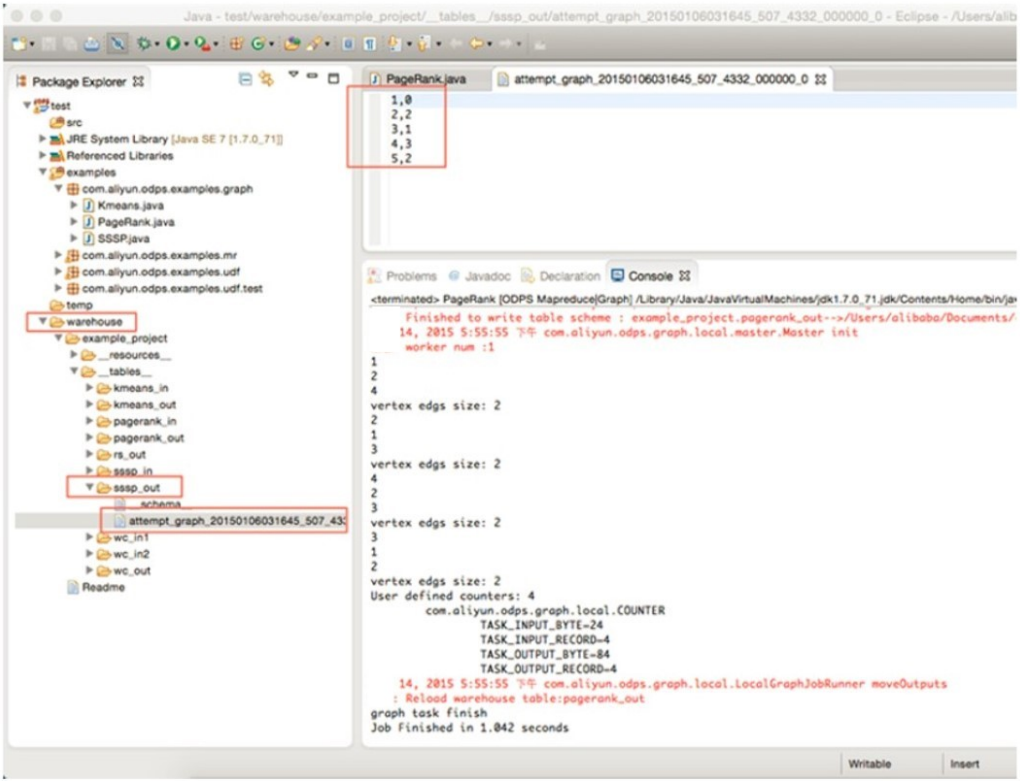
4. Click **Finish**. Check the running result, as shown in the following figure.

Running Result



Check the local computing result as shown below:

Local computing result



After passing the debugging, you can package the program, upload it to MaxCompute in the form of JAR resource, and submit the Graph job.

> **Note**
>
> - For more information about the packaging process, see MapReduce running example.
> - For more information about the directory structure of local results, see MapReduce running example.
> - For more information about uploading JAR resources, see Compile and run a Graph job.

1.22. MaxCompute FAQ

This topic describes MaxCompute FAQ and solutions.

SQL statement execution is slow. How do I check MaxCompute resource usage?

Log on to the MaxCompute AG as the admin user and perform the following steps:

1. Run the following command to display the remaining resources on the hosts in the MaxCompute cluster in ascending order:

```
r tfrl|sed 's/,//g'|sort -t "|" -k2 -n
```

2. Run the following command to view resource details of the hosts and the total cluster resources in MaxCompute:

```
r ttrl|sed 's/,//g'
```

3. Determine whether the remaining resources in MaxCompute are sufficient based on the ratio of remaining resources to total resources.

How do I handle the slow execution of jobs submitted by a project in a MaxCompute cluster with sufficient resources?

A possible cause is that the resources for the quota group where the project is located are exhausted. Perform the following steps to check whether the resources are exhausted and determine whether to add resources to the quota group:

1. Log on to the MaxCompute AG as the admin user and run the following command to check the resource usage of the quota group:

```
r quota
```

2. If you confirm that the resources for the quota group are exhausted, you can make modifications to the quota list on MaxCompute in Big Data Manager.

How do I modify quota group settings?

1. Run the following command in the MaxCompute AG to create or modify a quota:

```
sh/apsara/deploy/rpc_wrapper/rpc.sh setquota -i $QUOTAID -a $QUOTANAME -t fair -s$max_cpu_
quota $max_mem_quota -m $min_cpu_quota $min_mem_quota
```

> **Note** If $QUOTAID already exists, that quota is modified. Otherwise, a quota with that ID is created.

2. Log on to MaxCompute in Big Data Manager and configure relevant settings.

How do I perform simple operations on the metadata warehouse?

1. Log on to the MaxCompute AG.
2. Run the following commands:

```
/apsara/odps_tools/clt/bin/odpscmd
```

```
use meta;
```

3. Run the following command to view all tables in the metadata warehouse:

```
show tables;
```

4. Run the following command to obtain the description of a specific table:

```
desc <table>;
```

How do I use the smart metadata warehouse (package+view)?

You need to install the metadata warehouse enhancement package **package+view** first. Before you install the package, ensure that you are the owner of a project that is granted the package installation permission. After you have installed the package, you can follow the instructions provided in Package usage to use the smart metadata warehouse.

The following is a brief description on how to use the smart metadata warehouse:

- **Installation**

  package+view is a metadata warehouse enhancement package that depends on the metadata warehouse. You can install it by running the `odpscmd --config=odps_config.ini -f init.sql` command in the *system metadata warehouse* directory.

  > **Note**
  > - If the system asks you to re-install the package, remove the create package comment in the first line of init.sql and then run the preceding command again.
  > - **odps_config.ini**: This configuration file contains the configuration of the account used to access the metadata warehouse, which is also the project owner of the metadata warehouse.

- **Authorization**

  Run the following command to allow a project, such as p1, to install the package+view package:

  ```
  odpscmd --config=odps_config.ini -e "allow project p1 to install package systables;"
  ```

  Run the following command to allow all projects to install the package+view package:

  ```
  odpscmd --config=odps_config.ini -e "allow project * to install package systables;"
  ```

- **User operations**

  You can run the following command to install the package. Ensure that you are the owner of the project that is granted the package installation permission.

  ```
  install package meta.systables;
  ```

  After the installation is complete, you can run the following command to see the description of views in the package:

  ```
  desc package meta.systables;
  ```

  > **Note** After you have completed the preceding operations, you can start to use the smart metadata warehouse.

- **Views**

> **Note** To learn the definition of table schema, refer to the relevant content in the view description, which is available after you run the following command:
>
> ```
> desc viewname
> ```

View name	Content
allowed_package_installers	Information about the project that is granted the package installation permission
column_label_grants	Column label authorization information
column_labels	Column label information of a table
columns	Table schema information
installed_packages	Information about the package installed for the project
object_privileges	Table, UDF, resource authorization information
package_resources	Object information contained in the package
partitions	Partition information of a partitioned table
policies	User, role, and permission information defined in policies
resources	Resource information
roles	Role information
table_label_grants	Label authorization information of a table
table_labels	Label information of a table
tables	Table and view information
tasks	Job execution records
tunnels	Data upload and download records
udf_resources	Information about resources used in UDFs
udfs	Information about UDFs
user_roles	User and role association information
users	User information

- **Notes**

- By default, the smart metadata warehouse allows you to query data from the past 180 days. If you do not run a job on a specific day, you cannot obtain query results of that day from the smart metadata warehouse.
- We recommend that you specify a query range so that the system does not scan all data from the past 180 days.
- The time the data of the previous day is available depends on the specific output time of the metadata warehouse in the early morning. The data is available immediately after it is generated.
- The metadata warehouse does not provide metadata on the day when the project is created. The purpose is to avoid obtaining data of the project with the same name.

How do I grant Java sandbox permissions?

1. Log on to the AdminConsole and choose **MaxCompute Configuration > Project Management**. Select the project to which you want to grant Java sandbox permissions and double-click it to open the property dialog box.
2. In the **ODPS Sandbox Setting** area, enter the method or class you want to use in Sandbox Java Permissions.

   > **Note** Make sure that your input is in the correct format. The following example is for demonstration only. Enter each item in a single line and end it with a semicolon (;).

   ```
   permission java.lang.RuntimePermission "readSystemProperty"; permission java.lang.RuntimePermission "modifyThreadGroup"; permission java.security.AllPermission;
   ```

3. Click **Finish Modification**.

How do I handle exhausted disk capacity?

In most cases, you can clear scripts to free disk capacity. The most possible cause is that the root directory of MaxCompute AG or the /apsara directory occupies too much disk space. Therefore, you need to clear scripts in these two directories.

How do I find an AccessKey pair and configure correct AccessKey information?

1. Access the framework cluster management page of the Apsara Stack data center, and choose **Operations > Cluster Operations**. Select the cluster for which you want to find and configure the AccessKey pair, and access the cluster configuration page.
2. Double-click the kv.conf file in the file list to find the AccessKey information. You can also modify the information in this file and then save your modifications.

How do I add a MaxCompute host to a blacklist?

1. Log on to the Apsara AG as the admin user. Run the following command to enable the Fuxi blacklist function:

   ```
   r sgf fuximaster"{\"fuxi_Enable_BadNodeManager\":false}"
   ```

2. Run the following command to view the Fuxi blacklist:

```
/apsara/deploy/rpc_wrapper/rpc.shblacklist cluster get
```

3. Run the following command to add a MaxCompute host to the Fuxi blacklist:

```
/apsara/deploy/rpc_wrapper/rpc.shblacklist cluster add $hostname
```

4. Run the following command to view the Fuxi blacklist again and confirm that the host is added:

```
/apsara/deploy/rpc_wrapper/rpc.shblacklist cluster get
```

How do I export data from MaxCompute?

There are two methods to export data from MaxCompute: The first is to use the Tunnel command. The second is to configure synchronization tasks in DataWorks to export data from MaxCompute to other destinations.

How do I view the current MaxCompute version?

Run the following commands to view the MaxCompute version:

```
cat /apsara/odps_info/version|grep odps
```

```
cat /apsara/version
```

How do I restart MaxCompute services?

1. Run the following commands to save the configurations of resident MaxCompute services to a file. This configuration file is required when you restart MaxCompute services.

```
ssh odpsAG
cd /home/admin/
If you do not use a service, you can ignore the corresponding command.
You can use the r al command to view resident services.
r plan Odps/CGServiceControllerx > CGServiceControllerx
r plan sys/sqlonline-OTS >sqlonline-OTS
r plan Odps/MessagerServicex >MessagerServicex
r plan Odps/OdpsServicex >OdpsServicex
r plan Odps/HiveServerx >HiveServerx
r plan Odps/XStreamServicex >XStreamServicex
r plan Odps/QuotaServicex > QuotaServicex
r plan Odps/ReplicationServicex >ReplicationServicex
```

2. Run the following commands to stop MaxCompute services:

```
r sstop Odps/CGServiceControllerx
r sstop sys/sqlonline-OTS
r sstop Odps/MessagerServicex
r sstop Odps/OdpsServicex
r sstop Odps/HiveServerx
r sstop Odps/XStreamServicex
r sstop Odps/QuotaServicex
r sstop Odps/ReplicationServicex
```

3. Run the following commands to start MaxCompute services:

```
ssh odpsAG
cd /home/admin/
r start CGServiceControllerx
r start sqlonline-OTS
r start MessagerServicex.txt
r start OdpsServicex.txt
r start HiveServerx.txt
r start XStreamServicex.txt
r start QuotaServicex.txt
r start ReplicationServicex.txt
```

How do I power MaxCompute on and off?

1. Run the following commands to save the configurations of resident MaxCompute services to a file. This configuration file is required when you restart MaxCompute services.

```
ssh odpsAG
cd /home/admin/
If you do not use a service, you can ignore the corresponding command.
You can use the r al command to view resident services.
r plan Odps/CGServiceControllerx > CGServiceControllerx
r plan sys/sqlonline-OTS >sqlonline-OTS
r plan Odps/MessagerServicex >MessagerServicex
r plan Odps/OdpsServicex >OdpsServicex
r plan Odps/HiveServerx >HiveServerx
r plan Odps/XStreamServicex >XStreamServicex
r plan Odps/QuotaServicex > QuotaServicex
r plan Odps/ReplicationServicex >ReplicationServicex
```

2. Run the following commands to stop MaxCompute services:

```
r sstop Odps/CGServiceControllerx
r sstop sys/sqlonline-OTS
r sstop Odps/MessagerServicex
r sstop Odps/OdpsServicex
r sstop Odps/HiveServerx
r sstop Odps/XStreamServicex
r sstop Odps/QuotaServicex
r sstop Odps/ReplicationServicex
```

3. Run the following command to shut down the Apsara system:

```
/home/admin/dayu/bin/allapsara stop
```

4. Run the following command to shut down compute nodes gracefully:

```
Shutdown
```

5. Start compute nodes.
6. Run the following command to start the Apsara system:

```
/home/admin/dayu/bin/allapsara start
```

7. Run the following commands to start MaxCompute services:

```
ssh odpsAG
cd /home/admin/
r start CGServiceControllerx
r start sqlonline-OTS
r start MessagerServicex.txt
r start OdpsServicex.txt
r start HiveServerx.txt
r start XStreamServicex.txt
r start QuotaServicex.txt
r start ReplicationServicex.txt
```

How do I reduce the heavy load on a host?

1. Log on to the host with a heavy load and run the top command to check whether task processes occupy too many resources.
2. Generally, resources are occupied by user tasks. If task processes occupy too many resources, wait until these tasks are completed or ask users to stop these tasks.

1.23. Open source features of MaxCompute

This topic describes open source features related to MaxCompute.

SDKs

MaxCompute provides Java SDK and Python SDK interfaces to create, view, and delete MaxCompute tables. You can use the SDKs to manage MaxCompute by editing code.

How to obtain service support: Visit the official documentation or submit a ticket online.

MaxCompute RODPS

MaxCompute RODPS is an R plug-in for MaxCompute. For more information about the plug-in, see ODPS Plugin for R on GitHub.

How to obtain service support: Leave a message or create an issue in ODPS Plugin for R on GitHub.

MaxCompute JDBC

MaxCompute JDBC is an official JDBC driver provided by MaxCompute. It provides a set of interfaces to execute SQL tasks for Java programs. The project is hosted in ODPS JDBC on GitHub.

How to obtain service support: Leave a message or create an issue in ODPS JDBC on GitHub.

Mars

Mars is a tensor-based unified distributed computing framework. Mars makes it possible to execute large-scale scientific computing tasks by using only several lines of code, whereas MapReduce requires hundreds of lines of code. In addition, Mars improves computing performance.

The source code of Mars is now available on GitHub. You are welcome to contribute to Mars. You can visit Mars on GitHub to obtain its open source code.

For more information about Mars, see Mars Development Guide.

How to obtain service support: Leave a message or create an issue in Mars on GitHub.

Data Collector

Data Collector is a collection of the major open source data collection tools of MaxCompute, such as the Flume plug-in, OGG plug-in, Sqoop, Kettle plug-in, and Hive Data Transfer UDTF.

The Flume and OGG plug-ins are implemented based on the DataHub SDK, whereas Sqoop, the Kettle plug-in, and Hive Data Transfer UDTF are implemented based on the Tunnel SDK. DataHub is a real-time data transfer channel, and Tunnel is a batch data transfer channel. The Flume and OGG plug-ins are used to transfer data in real time. Sqoop, the Kettle plug-in, and Hive Data Transfer UDTF are used to transfer data in batches in offline mode.

For information about the source code, see Aliyun MaxCompute Data Collectors on GitHub. For more information about these tools, see wiki on GitHub.

How to obtain service support: Leave a message or create an issue in Aliyun MaxCompute Data Collectors on GitHub.

2.DataWorks

2.1. Log on to the DataWorks console

This topic describes how to log on to the DataWorks console.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

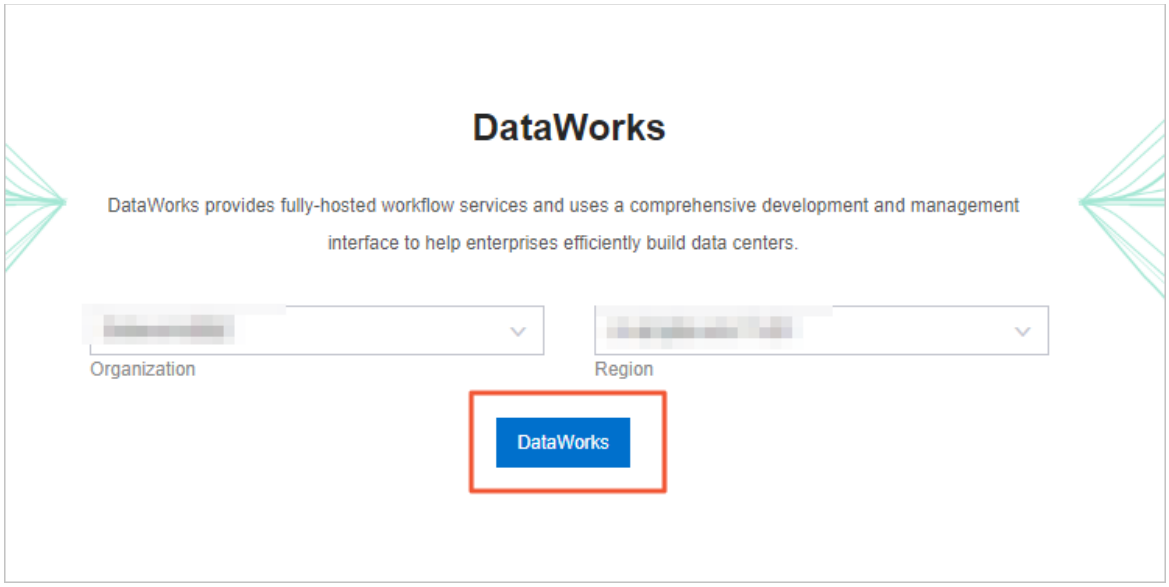
Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products > Big Data > DataWorks**.
5. On the page that appears, set the **Organization** and **Region** parameters and click **DataWorks**.



Notice You cannot log on to the DataWorks console by using the root organization.

2.2. Create a workspace

This topic describes how to create a workspace on the Project Management page.

Prerequisites

A compute engine is created to initialize MaxCompute projects.

Overview

DataWorks provides various preset templates for a workspace administrator to select when the administrator creates workspaces that contain one or more working environments, including development, testing, staging, and production. DataWorks can also automatically generate associations between workspaces. A one-to-many relationship exists between departments and workspaces. That is, multiple workspaces can be created under a department.

You can create a workspace in one of the following modes:

- **Standard Mode (Development and Production Environments)**: A DataWorks workspace corresponds to two MaxCompute projects. One MaxCompute project serves as the development environment and the other serves as the production environment.
- **Basic Mode (Production Environment Only)**: A DataWorks workspace corresponds to only one MaxCompute project.

Note For more information about the two workspace modes, see Workspace modes.

Procedure

1. Log on to the DataWorks console as a workspace administrator.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Project Management**. By default, the **Workspaces** page appears.
3. On the Workspaces page, click **Create Workspace** in the upper-right corner.

4. In the **Create Workspace** dialog box that appears, set the parameters in the Basic Information section.

   > **Note** If you select the standard mode, you must associate the workspace with two MaxCompute projects.

5. Set the parameters in the **Advanced Settings** section. You can select whether to enable the recurrence and whether to allow downloading the query results returned by SELECT statements. You must associate the workspace with MaxCompute projects.
6. Click **OK**.

2.3. Quick Start

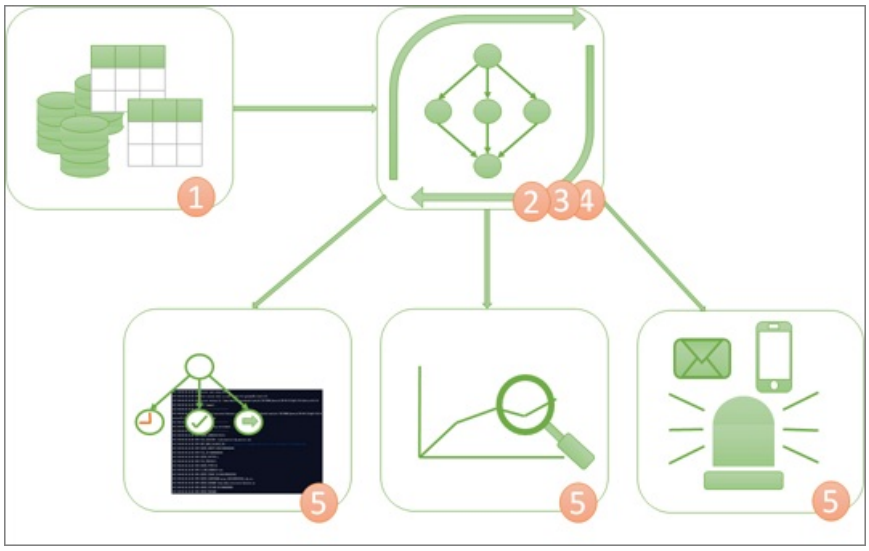
2.3.1. Overview

Quick Start guides you through a complete process of data analytics and O&M.

Generally, you can complete the following data analytics and O&M operations in a workspace of DataWorks:

1. Create tables and import data.
2. Create a workflow.
3. Create a sync node.
4. Configure recurrence and dependencies for a node.
5. Run a node and troubleshoot errors.

The following figure shows the basic process of data analytics and O&M.



2.3.2. Create tables and import data

This topic takes the bank_data and result_table tables as an example to describe how to create tables and import data in the DataWorks console.

Note The bank_data table stores business data, whereas the result_table table stores data analytics results.

Create the bank_data table

1. Log on to the DataWorks console.
2. On the **DataStudio** page that appears, move the pointer over the icon and click **Table**.
3. In the **Create Table** dialog box, set **Table Name** to bank_data.
4. Click **Commit**.
5. On the editing page of the created table, click **DDL Statement**.
6. In the **DDL Statement** dialog box, enter the table creation statement, and click **Generate Table Schema**. In the dialog box that appears, click **OK**.

   In this topic, the following statement is used as an example:

```
CREATE TABLE IF NOT EXISTS bank_data
(
 age           BIGINT COMMENT 'age',
 job           STRING COMMENT 'job type',
 marital       STRING COMMENT 'marital status',
 education     STRING COMMENT 'education level',
 default       STRING COMMENT 'credit card',
 housing       STRING COMMENT 'mortgage',
 loan          STRING COMMENT 'loan',
 contact       STRING COMMENT 'contact',
 month         STRING COMMENT 'month',
 day_of_week   STRING COMMENT 'day in a week',
 duration      STRING COMMENT 'duration',
 campaign      BIGINT COMMENT 'number of contacts during the campaign',
 pdays         DOUBLE COMMENT 'interval from the last contact',
 previous      DOUBLE COMMENT 'number of contacts with the customer',
 poutcome      STRING COMMENT 'result of the previous marketing campaign',
 emp_var_rate  DOUBLE COMMENT 'employment change rate',
 cons_price_idx DOUBLE COMMENT 'consumer price index',
 cons_conf_idx DOUBLE COMMENT 'consumer confidence index',
 euribor3m     DOUBLE COMMENT 'Euro deposit rate',
 nr_employed   DOUBLE COMMENT 'number of employees',
 y             BIGINT COMMENT 'whether time deposit is available'
);
```

7. After the table schema is generated, enter the display name of the table and click **Commit to Development Environment** or **Commit to Production Environment**.

> Note If you are using a workspace of the basic mode, click Commit to Production Environment.

8. In the left-side navigation pane, click **Workspace Tables**. On the page that appears, enter the table name to search for the created table. After you find the table, double-click the table name to view the table information.

Create the result_table table

1. On the **DataStudio** page that appears, move the pointer over the icon and click **Table**.
2. In the **Create Table** dialog box, set **Table Name** to result_table.
3. On the editing page of the created table, click **DDL Statement**.
4. In the **DDL Statement** dialog box, enter the table creation statement, and click **Generate Table Schema**. In the dialog box that appears, click **OK**.

   In this topic, the following statement is used as an example:

```
CREATE TABLE IF NOT EXISTS result_table
(
 education      STRING COMMENT 'education level',
 num        BIGINT COMMENT 'number of people'
);
```

5. After the table schema is generated, enter the display name of the table and click **Commit to Development Environment** or **Commit to Production Environment**.
6. In the left-side navigation pane, click **Workspace Tables**. On the page that appears, enter the table name to search for the created table. After you find the table, double-click the table name to view the table information.

Upload a local file to import its data to the bank_data table

You can perform the following operations in the DataWorks console:

- Upload a local text file to import its data to a table in a workspace.
- Use Data Integration to import business data from different data stores to a workspace.

> Note In this topic, a local file is used as the source of data. Comply with the following rules when uploading a local file:
>
> - File format: The file must be in the .txt, .csv, or .log format.
> - File size: The size of the file cannot exceed 10 MB.
> - Destination object: The destination object can be a partitioned table or a non-partitioned table. The partition key value cannot be in Chinese.

To upload the local file banking.txt to DataWorks, follow these steps:

1. On the **Data Analytics** tab, click the **Import** icon.
2. In the **Data Import Wizard** dialog box, select the table to which you want to import data and click **Next**.

3. Set **Select Data Import Method** to **Upload Local File** and click **Browse**. In the dialog box that appears, select the target local file and configure import information.

Parameter	Description
Select Data Import Method	The method of importing data. Valid values: **Upload Local File**, **DataService Studio**, and **Workbooks from Data Analysis**. In this example, select **Upload Local File**.
Select File	Click **Browse** and select the local file to upload.
Select Delimiter	The delimiter of fields in the file. Valid values: **Comma**, **Tab**, **Semicolon**, **Space**, **|**, **#**, and **&**. In this example, select **Comma**.
Original Character Set	The character set of the file. Valid values: **GBK**, **UTF-8**, **CP936**, and **ISO-8859**. In this example, select **GBK**.
Import First Row	The line from which data is to be imported. In this example, select **1**.
First Row as Field Names	Specifies whether to use the first line as the header line.
Data Preview	The preview of the data to import. **Note** If the data volume is large, only the data in the first 100 lines and 50 columns appears.

4. After the configuration is completed, click **Next**.
5. Select a matching mode for the fields in the source file and destination table. In this example, select **By Location**.
6. Click **Import Data**.

Data import methods

- Create a sync node

  This method is used to import data from various data stores, such as Relational Database Service (RDS), MySQL, SQL Server, PostgreSQL, MaxCompute, ApsaraDB for Memcache, Distribute Relational Database Service (DRDS), Object Storage Service (OSS), Oracle, FTP, DM, Hadoop Distributed File System (HDFS), and MongoDB.

- Upload a local file

  This method is used to upload .txt and .csv files not exceeding 10 MB. The destination object can be a partitioned table or a non-partitioned table. The partition key value cannot be in Chinese.

- Run Tunnel commands to upload a file

  This method is used to upload local files and other resource files of any size.

What to do next

Now you have learned how to create tables and import data. You can proceed with the next tutorial. In the next tutorial, you will learn how to create a workflow and how to compute and analyze data in a workspace. For more information, see Create a workflow.

2.3.3. Create a workflow

This topic describes how to create a workflow, create nodes in the workflow, and configure the dependencies among the nodes. After the configuration is completed, you can use the Data Analytics feature to further compute and analyze data in the workspace.

Prerequisites

The bank_data table for storing business data and the result_table table for storing data analytics results are created in the workspace. Data is imported to the bank_data table. For more information, see Create tables and import data.

Context

The Data Analytics feature of DataWorks allows you to drag and drop nodes in a workflow and configure the dependencies among the nodes. You can process data and configure dependencies in the data based on the workflow.

Create a workflow

1. Log on to the DataWorks console.
2. On the **DataStudio** page that appears, move the pointer over the **Create** icon and click **Workflow**.
3. In the **Create Workflow** dialog box, set the **Business Name** and **Description** parameters.
4. Click **Create**.

Create nodes and configure dependencies among the nodes

This section describes how to create a zero load node named start and an ODPS SQL node named insert_data in the workflow, and configure the insert_data node to depend on the start node.

> **Note** Pay attention to the following points when you use a zero load node:
>
> - A zero load node is a control node used to maintain and control its descendant nodes. When the zero load node runs in a workflow, it does not generate any data.
> - If other nodes are dependent on the zero load node and it is manually set to Failed by an administration expert, the pending descendant nodes cannot be triggered. During the O&M process, an administration expert can disable the zero load node to prevent errors of ancestor nodes from being further expanded.
> - Typically, the ancestor node of the zero load node in a workflow is set to the root node of the workspace. The root node of the workspace is named in the `Workspace name_root` format.

We recommend that you create a zero load node as the root node of a workflow to control the entire workflow.

1. Double-click the name of the workflow to go to the dashboard of the workflow. Move the

pointer over **Zero-Load Node** and drag it to the development panel on the right.

2. In the **Create Node** dialog box, set **Node Name** to start and click **Commit**.
3. Repeat steps 1 and 2 to create an **ODPS SQL** node and name it insert_data.
4. Draw a line to connect the nodes and set the start node as the ancestor node of the insert_data node.

Configure the ancestor node of the zero load node

The zero load node in a workflow is the controller of the entire workflow, and also the ancestor of all nodes in the workflow. Generally, the zero load node in a workflow depends on the root node of the workspace.

1. Double-click the name of the zero load node. On the page that appears, click the **Properties** tab in the right-side navigation pane.
2. In the **Properties** section, click **Use Root Node** and set the ancestor node of the zero load node as the root node of the workspace.
3. After the configuration is completed, click  in the upper-left corner.

Edit code in the ODPS SQL node

This section provides a sample SQL statement used to query and save the number of singles with different education levels who loan to buy houses in the ODPS SQL node insert_data. The queried data can be analyzed by and presented in descendant nodes of insert_data.

The SQL statement is as follows:

```
INSERT OVERWRITE TABLE result_table  --Insert data to the result_table table.
SELECT education
   , COUNT(marital) AS num
FROM bank_data
WHERE housing = 'yes'
   AND marital = 'single'
GROUP BY education
```

Run and debug the ODPS SQL node

1. After the SQL statement is entered in the insert_data node, click **Save**.
2. Click **Run** to view the runtime logs and result.

Commit the workflow

1. After running and debugging the ODPS SQL node insert_data, return to the workflow editing page and click **Commit**.
2. In the **Commit** dialog box, select the nodes to be committed, set **Description**, and then select **Ignore I/O Inconsistency Alerts**.
3. Click **Commit**.

What to do next

Now you have learned how to create and commit a workflow. You can proceed with the next tutorial. In the next tutorial, you will learn how to create a sync node to export data to different types of data stores. For more information, see Create a sync node.

2.3.4. Create a sync node

This topic describes how to create a sync node to export data from MaxCompute to a MySQL database.

Background

In DataWorks, Data Integration can be used to periodically transfer the business data generated in a business system to a workspace. After the data is computed in SQL nodes, Data Integration periodically exports the computing results to your specified data store for further display or use.

Add a connection

> **Note** Only the workspace administrator can create connections, and members of other roles can only view the connections.

1. Log on to the DataWorks console. On the DataSudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration**.
2. In the left-side navigation pane, choose **Sync Resources > Connections**. On the **Connections** page, click **Create Connection**.
3. In the **Create Connection** dialog box, set Data Source Type to **MySQL**.
4. Set parameters in the **Add MySQL Connection** dialog box. The following table lists the parameters that need to be set when the connection type is set to **User-Created Data Store**.

Parameter	Description
Data Source Type	The type of the connection. In this example, set the type to **MySQL > User-Created Data Store**.
Data Source Name	The name of the connection. The name can contain letters, digits, and underscores (_) and must start with a letter.
Description	The description of the connection. The description cannot exceed 80 characters in length.
Applicable Environment	The environment where the connection is configured. Valid values: **Development** and **Production**. **Note** This field is available only for workspaces in standard mode.
JDBC URL	The JDBC connectivity URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.

Parameter	Description
Username	The username for logging on to the database. Note You must enter the information of your MySQL database.
Password	The password for logging on to the database.

5. Click **Test Connection**.
6. If the connectivity test is successful, click **Complete**.

Verify that a table exists in the destination MySQL database

Use the following table creation statement to create the odps_result table in the MySQL database:

```
CREATE TABLE `ODPS_RESULT` (
`education` varchar(255) NULL ,
`num` int(10) NULL
);
```

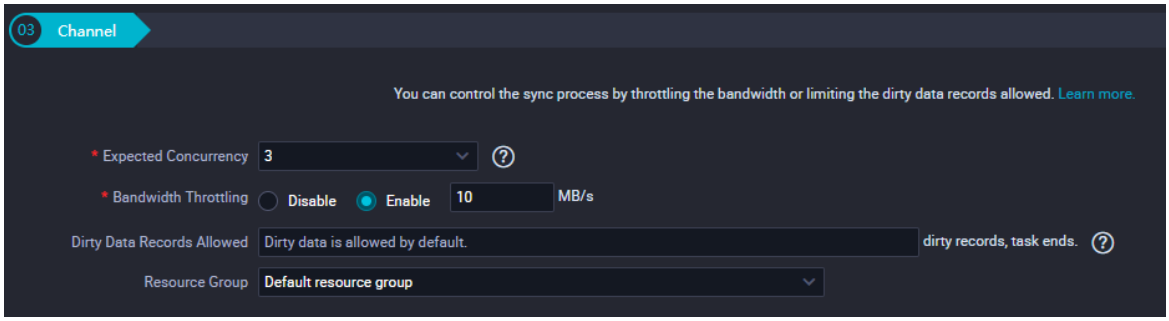
After the table is created, run the `desc odps_result;` statement to view the table details.

Create and configure a sync node

This section describes how to create and configure the sync node write_result to export data in the result_table table to your MySQL database. The procedure is as follows:

1. Go to the **Data Analytics** tab and create the sync node write_result.
2. Configure the insert_data node as the ancestor node of the write_result node.
3. Set **Data Source** to **MaxCompute > odps_first** and **Table** to result_table.
4. Select the odps_result table in your MySQL database as the destination table.
5. Configure the mapping between the fields in the source and destination tables.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add Line** to add a field or move the pointer over a field and click the **Delete** icon to delete the field.
6. In the Channel section, configure the synchronization rate limit and dirty data check rules.



Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The servers on which nodes are run. If an excessively large number of nodes are run in the default resource group, some nodes may be delayed due to insufficient resources. In this case, we recommend that you add a custom resource group.

7. Preview and save the configuration.

   After the configuration is completed, scroll up and down to view the node configuration. Verify that the configuration is correct and click **Save**.

Commit the sync node

Return to the workflow after saving the sync node. In the top navigation bar, click **Commit** to commit the sync node to scheduling system. The scheduling system automatically and periodically runs the node starting from the next day based on the configured properties of the node.

What to do next

Now you have learned how to create a sync node to export data to a specific data store. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure recurrence and dependencies for a sync node. For more information, see Configure recurrence and dependencies for a node.

2.3.5. Configure recurrence and dependencies for a node

This topic describes how to configure recurrence and dependencies for a node in the DataWorks console.

> **Note** In this topic, the sync node write_result is used as an example and the recurrence is set to weekly.

DataWorks has a powerful scheduling engine to trigger nodes based on the recurrence and dependencies of nodes. DataWorks guarantees that tens of millions of nodes run accurately and punctually per day based on directed acyclic graphs (DAGs). In the DataWorks console, you can set the recurrence to minutely, hourly, daily, weekly, or monthly.

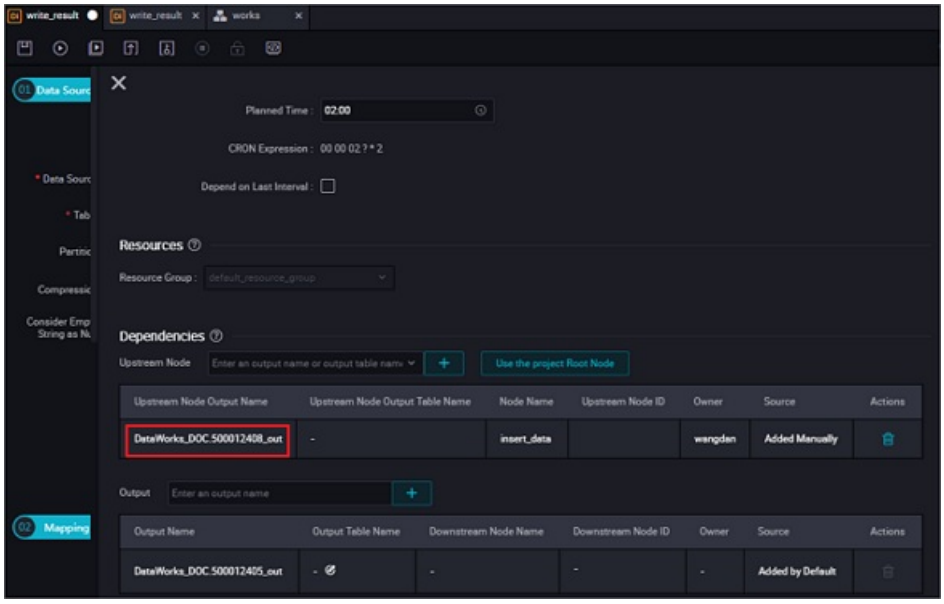
Configure recurrence for the sync node

1. After the sync node write_result is created, double-click the sync node to configure it.
2. Click the **Properties** tab in the right-side navigation pane to configure recurrence for the sync node.

Parameter	Description
Execution Mode	The mode in which the node is run. Valid values: **Normal** and **Dry-Run**. You can select one based on your own needs.
Retry Upon Error	Specifies whether to rerun the node upon an error.
Valid From	The date from which the node is effective.
Skip Execution	Specifies whether to skip execution of the node.
Cycle	The recurrence of the node, which can be monthly, weekly, daily, hourly, or minutely. In this example, the recurrence is set to weekly.
Customize Runtime	Specifies whether to run the node periodically. This field is selected by default.
Run Every or **Run At**	The specific day or time when the node is run. For example, you can configure a node to run at 02:00 every Tuesday.
CRON Expression	The value is `00 00 02 ? * 2` by default. It cannot be modified.
Cross-Cycle Dependencies	Specifies whether the node depends on the result of the last cycle.

Configure dependencies for the sync node

After configuring recurrence for the sync node write_result, you can continue to configure dependencies for the sync node.



You can configure the ancestor node on which the sync node depends. After that, the scheduling system can trigger the sync node when the specified time arrives, only after the instance of the ancestor node is run.

The configuration shown in the preceding figure indicates that the instance of the sync node is not triggered until the instance of the ancestor node insert_data is run.

The scheduling system creates the Workspace name_root node for each workspace as the root node by default. If no ancestor node is configured for the sync node, the sync node depends on the root node.

Commit the sync node

Save the configuration of the sync node write_result and click **Commit** to commit the node to the scheduling system.

Only after a node is committed, the scheduling system can automatically generate and run instances at the specified time starting from the next day according to the recurrence property.

> **Note** If a node is committed after 23:30, the scheduling system automatically generates and runs instances of the node starting from the third day.

What to do next

Now you have learned how to configure recurrence and dependencies for a sync node. You can proceed with the next tutorial. In the next tutorial, you will learn how to perform O&M on the committed node and troubleshoot errors based on the runtime logs. For more information, see Run a node and troubleshoot errors.

2.3.6. Run a node and troubleshoot errors

This topic describes how to run and maintain a node, and troubleshoot errors based on logs.

When you configure recurrence and dependencies for the sync node write_result, you have configured the sync node to run at 02:00 every Tuesday. After you commit this node, you have to wait until the next day to view the automatic execution result of this node. DataWorks allows you to run nodes in the following modes: test run, retroactive run, and periodic run. This helps you confirm the run time of each node instance, dependencies among node instances, and whether generated data meets your expectation.

- Test run: Nodes are triggered manually. This method is recommended if you only want to confirm the run time and running of a single node.
- Retroactive run: Nodes are triggered manually. This method is recommended if you want to confirm the run time of multiple nodes and dependencies among them, or if you want to re-perform data analysis and computing from the specific root node.
- Periodic run: Nodes are triggered automatically. The scheduling system automatically triggers the instances of committed nodes at the specified time points starting from 00:00 the next day after the nodes are committed. In addition, the scheduling system checks whether the ancestor instances of each instance have been run when the scheduled time arrives. If all the ancestor instances have been run when the scheduled time arrives, the current instance is automatically triggered without manual intervention.

Note The scheduling system generates instances for manually triggered nodes and auto triggered nodes based on the same rules.

- The scheduling system generates an instance for each recurrence, which can occur by day, hour, minute, month, or week.
- The scheduling system runs an instance only on the specified date and generates runtime logs for the instance.
- The scheduling system does not run an instance on other dates except the specified date. Instead, it directly change the status of the instance to successful when the running conditions are met. In this case, the scheduling system does not generate runtime logs.

Test run

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products** > **Operation Center** to go to the **Operation Center** page.
3. In the left-side navigation pane, click **Recurring**. On the page that appears, find the target node to run. Click **Test** next to the target node.
4. In the **Smoke Test** dialog box, set the **Smoke Test Instance Name** and **Data Timestamp** parameters and click **OK**.
5. On the **Smoke Test** page that appears, click an instance. The directed acyclic graph (DAG) of the instance appears on the right.

   Right-click the instance to view its dependencies and details, and stop or re-run this instance.

   Note
   - In test run mode, a node is triggered manually. The corresponding instance runs immediately when the scheduled time arrives, regardless of whether its ancestor instances have been run.
   - The sync node write_result is configured to run at 02:00 every Tuesday. According to the instance generation rules described earlier in this topic, if the data timestamp, which is one day before the run date, is set to Monday for a test run, the scheduling system runs the instance for the sync node write_result at 02:00 on Tuesday. If the data timestamp is not set to Monday for the test run, the scheduling system changes the status of the instance to successful at 02:00 on Tuesday with no runtime logs generated.

Retroactive run

A retroactive run is recommended if you want to confirm the run time of multiple nodes and dependencies among them, or if you want to re-perform data analysis and computing from the specific root node.

1. On the **Operation Center** page, choose **Task List** > **Recurring** in the left-side navigation pane.
2. Find the target node to run and choose **Patch Data** > **Current Node Retroactively** for the

target node.

3. In the **Patch Data** dialog box, set parameters and click **OK**.

Parameter	Description
Retroactive Instance Name	Enter the name of the retroactive instance.
Data Timestamp	Select the data timestamp of the retroactive instance. The retroactive instance is run on the next day of the specified timestamp.
Node	The default value is the current node, which cannot be changed.
Parallelism	Select **Disable** or specify several nodes to run concurrently.

4. On the **Retroactive** page that appears, click the retroactive instance to view the DAG of the instance.

   Right-click the instance to view its dependencies and details, and stop or re-run this instance.

   > **Note**
   > - In retroactive run mode, instance running requires the result of instance running on the previous day. For example, retroactive instances are configured to run between September 15, 2017 and September 18, 2017. If the instance on September 15 fails to run, the instance on September 16 cannot run.
   > - The sync node write_result is configured to run at 02:00 every Tuesday. According to the instance generation rules described earlier in this topic, if the data timestamp, which is one day before the run date, is set to Monday for a retroactive instance, the scheduling system runs the instance for the sync node write_result at 02:00 on Tuesday. If the data timestamp is not set to Monday for the retroactive instance, the scheduling system changes the status of the instance to successful at 02:00 on Tuesday with no runtime logs generated.

Periodic run

In periodic run mode, the scheduling system automatically triggers instances for all nodes based on the scheduling configuration. No menu item is provided for you to control the periodic run on the DataStudio page. You can view the instance information and runtime logs in either of the following ways:

- On the **Operation Center** page, choose **Operation Center** > **Node O&M** > **Recurring** in the left-side navigation pane. On the page that appears, set parameters such as the data timestamp or run date, find an instance of the sync node write_result, and then right-click the instance to view the instance information and runtime logs.
- On the **Recurring** page, click the instance of the target node to view the DAG of the instance.

  Right-click the instance to view its dependencies and details, and stop or re-run this instance.

> Note
>
> - If an ancestor node is not run, a descendant node does not run either.
> - If the initial status of an instance is pending, the scheduling system checks whether all its ancestor instances have been run when the scheduled time arrives.
> - The instance can be triggered and run only after all its ancestor instances have been run and the scheduled time arrives.
> - If an instance is pending, check whether all its ancestor instances have been run and whether the scheduled time arrives.

2.4. Data Integration

2.4.1. Overview

Data Integration is a stable, efficient, and scalable data synchronization service. It is designed to migrate and synchronize data between a wide range of heterogeneous data stores fast and stably in complex network environments.

Limits

- Data Integration can synchronize structured, semi-structured, and unstructured data. Structured data stores include Relational Database Service (RDS) and Distributed Relational Database Service (DRDS). Unstructured data, such as Object Storage Service (OSS) objects and text files, must be capable of being converted to structured data. Data Integration can only synchronize data that can be abstracted to two-dimensional logical tables to MaxCompute. It cannot synchronize unstructured data that cannot be converted to structured data, such as MP3 files stored in OSS, to MaxCompute.
- Data Integration supports data synchronization and exchange in one region or between regions.

  Data can be transmitted between regions over the classic network, but the network connectivity is not guaranteed. If the transmission fails over the classic network, we recommend that you use an Internet connection.
- Data Integration supports only data synchronization but not data consumption.

Batch data synchronization

Data Integration can be used to synchronize large amounts of data. Data Integration facilitates data transmission between diverse structured and semi-structured data stores. It provides readers and writers for the supported data stores and defines a transmission channel between the source and destination data stores and datasets, based on simplified data types.

Supported data stores

- Relational databases: MySQL, SQL Server, PostgreSQL, Oracle, Dameng, DRDS, PolarDB, HybridDB for MySQL, AnalyticDB for PostgreSQL, AnalyticDB for MySQL 2.0, and AnalyticDB for MySQL 3.0
- Big data storage: MaxCompute, DataHub, and Data Lake Analytics (DLA)
- Semi-structured storage: OSS, Hadoop Distributed File System (HDFS), and FTP
- NoSQL: MongoDB, Memcache, Redis, and Tablestore

- Message queue: LogHub
- Graph compute engine: GraphCompute

For more information, see Supported data sources.

> Note The connection configurations for data stores vary greatly. You can view the specific parameters that need to be set when you configure connections and sync nodes for data stores.

Development modes of sync nodes

You can develop sync nodes in one of the following modes:

- **Codeless UI**: Data Integration provides step-by-step instructions to help you configure a sync node. This mode is easy to use but provides only limited features.
- **Code editor**: You can write a JSON script to create a sync node. This mode supports advanced features to facilitate flexible configuration. It is suitable for experienced users and increases the cost of learning.

> Note
> - The code generated for a sync node on the codeless user interface (UI) can be converted to a script. This conversion is irreversible.
> - Before you write code, you must configure a connection and create the destination table.

Network types

A data store can reside on the classic network or in a virtual private cloud (VPC). The user-created IDC network type has been planned and will be supported soon.

- Classic network: a network deployed by Alibaba Cloud, which is shared with other tenants. This network is easy to use.
- VPC: a network created on Alibaba Cloud, which is available to only one Apsara Stack account. You have full control over your VPC, including customizing the IP address range, dividing the VPC to multiple subnets, and configuring routing tables and gateways.

  A VPC is an isolated network for which you can customize a wide range of parameters, such as the IP address range, subnets, and gateways. Based on wide deployment of VPCs, Data Integration provides the feature to automatically detect the reverse proxy for some data stores, including ApsaraDB RDS for MySQL, ApsaraDB RDS for PostgreSQL, ApsaraDB RDS for SQL Server, PolarDB, DRDS, HybridDB for MySQL, AnalyticDB for PostgreSQL, and AnalyticDB for MySQL 3.0. By using this feature, you do not need to purchase an extra Elastic Compute Service (ECS) instance in your VPC to configure sync nodes for these data stores. Instead, Data Integration automatically uses this feature to provide network connectivity to these data stores.

  When you configure sync nodes for other Alibaba Cloud data stores in a VPC, such as PPAS, ApsaraDB for OceanBase, ApsaraDB for Redis, ApsaraDB for MongoDB, ApsaraDB for Memcache, Tablestore, and ApsaraDB for HBase, you must purchase an ECS instance in the same VPC. This ECS instance is used to access the data stores.
- User-created IDC network: an IDC network deployed by yourself, which is isolated from the

Alibaba Cloud network.

> **Note** You can access data stores over the Internet. However, the access speed depends on the Internet bandwidth, and additional network access expenses are required. We recommend that you do not use Internet connections.

Terms

- Concurrency

  Concurrency indicates the maximum number of concurrent threads to read data from or write data to data storage within a single sync node.

- Bandwidth throttling

  Bandwidth throttling indicates that a maximum transmission rate is specified for a sync node of Data Integration.

- Dirty data

  Dirty data indicates meaningless data and data that does not match the specified data type. For example, you want to write data of the VARCHAR type in the source table to an INT-type field in the destination table. A data conversion error occurs and the data cannot be written to the destination table. In this case, the data is dirty.

- Connection

  A connection in DataWorks is used for accessing a data store, which can be a database or a data warehouse. DataWorks supports various types of data stores, and supports data synchronization between data stores of different types.

2.4.2. Homepage

The Data Integration homepage provides entries for you to create sync nodes, manage connections, maintain sync nodes, and view help documents.

Log on to the DataWorks console, click  in the upper-left corner, and choose **All Products** > **Data Aggregation** > **Data Integration**. The homepage of Data Integration appears by default.

On this page, you can perform the following operations:

- **New Task**: Click here to go to the **Data Analytics** page, where you can create sync nodes. For more information, see Create a sync node.
- **Connection**: Click here to go to the **Data Source** page, where you can view created connections and add a connection or multiple connections at a time.
- **Workbench**: Click here to go to the **Operation Center** > **Dashboard** page, where you can view the running status of created nodes. For more information, see O&M Overview.

2.4.3. Connectivity testing

This topic describes the FAQ about connectivity testing on connections.

When configuring a security group for a connection hosted on an Elastic Compute Service (ECS) instance, add the IP address of the scheduling cluster to the inbound and outbound rules of the security group. If the security group is not properly configured, data synchronization fails due to a connection failure.

To set a wide port range for a security group rule, call relevant API operations, instead of using the console.

Common scenarios of connectivity test failures

When a connection fails the connectivity test, check whether the region, network type, whitelist, database name, and username are properly configured for the connection. The following errors may occur during connectivity testing:

- The database password is incorrect.
- The network connection fails.
- A network error occurs during data synchronization.

  Check the log and determine which resource group is used. Check whether the resource group is a custom one.

  For a Relational Database Service (RDS) connection or a MongoDB connection, if a custom resource group is used, check whether its IP addresses are added to the whitelist of the connection.

  Check whether both the source and destination connections pass the connectivity test. For an RDS connection or a MongoDB connection, check whether all relevant IP addresses are added to the whitelist of the connection. If the IP address of a server is not added to the whitelist, the sync node fails when it runs on this server. However, the sync node succeeds when it runs on another server whose IP address is added to the whitelist.
- The result shows that a sync node is run but the log contains a disconnection error in port 8000.

  This issue occurs because a custom resource group is used and no inbound rule is configured for the corresponding IP address and port 8000 in the security group. To resolve the issue, add the IP address and port to the inbound rule of the security group and run the node again.

Examples of connectivity test failures

Example 1

- Symptom

  A connection failed the connectivity test. The database connection failed. The following information is involved: Database URL: jdbc:mysql://xx.xx.xx.x:xxxx/t_uoer_bradef. Username: xxxx_test. Error message: Access denied for user 'xxxx_test'@'%' to database 'yyyy_demo'.
- Troubleshooting
  i. Check whether the configuration of the connection is correct.
  ii. Check whether the database password is correct, the whitelist is properly configured, and your account has the permission to access the database. You can grant the required permissions in the RDS console.
- Example 2

- Symptom

  A connection failed the connectivity test. The following error message is returned:

  ```
  error message: Timed out after 5000 ms while waiting for a server that matches ReadPreferenceServerSelector{readPreference=primary}. Client view of cluster state is {type=UNKNOWN, servers=[(xxxxxxxxxx), type=UNKNOWN, state=CONNECTING, exception={com.mongodb.MongoSocketReadException: Prematurely reached end of stream}}]
  ```

- Troubleshooting

  Before testing the connectivity to a MongoDB connection that is not deployed in a Virtual Private Cloud (VPC), add relevant IP addresses to the whitelist of the connection.

2.4.4. Data sources

2.4.4.1. Supported data stores and plug-ins

Data Integration is a stable, efficient, and scalable data synchronization service. It provides transmission channels for batch data stored in Alibaba Cloud services such as MaxCompute, AnalyticDB for PostgreSQL, and Hologres.

The following table lists the data stores and plug-ins that Data Integration supports.

Data store	Reader	Writer
ApsaraDB for OceanBase	ApsaraDB for OceanBase	ApsaraDB for OceanBase
DataHub	DataHub Reader	DataHub Writer
Db2	DB2 Reader	DB2 Writer
DM	RDBMS Reader	RDBMS Writer
DRDS	DRDS Reader	DRDS Writer
Elasticsearch	Elasticsearch Reader	Elasticsearch Writer
FTP	FTP	FTP Writer
GBase8a	Supported	GBase8a Writer
HBase	HBase Reader	• HBase Writer • HBase11xsql Writer
HDFS	HDFS Reader	HDFS Writer
Hive	Hive Reader	Hive Writer
Hologres	Supported	Supported
HybridDB for MySQL	Supported	Supported

Data store	Reader	Writer
LogHub	LogHub Reader	LogHub Writer
MaxCompute	MaxCompute Reader	MaxCompute Writer
Memcache	Not supported	Memcache Writer
MongoDB	MongoDB Reader	MongoDB Writer
MySQL	MySQL Reader	MySQL Writer
Oracle	Oracle Reader	Oracle Writer
OSS	OSS Reader	OSS Writer
POLARDB	Supported	Supported
PostgreSQL	PostgreSQL Reader	PostgreSQL Writer
RDBMS	RDBMS Reader	RDBMS Writer
Redis	Not supported	Redis Writer
Stream	Stream Reader	Stream Writer
SQL Server	SQL Server Reader	SQL Server Writer
Tablestore	Tablestore Reader	Tablestore Writer
Vertica	Vertica Reader	Vertica Writer

2.4.4.2. Connection isolation

DataWorks provides the connection isolation feature to isolate data of the development environment from that of the production environment for workspaces in standard mode.

If a connection is configured in both the development and production environments, you can use the connection isolation feature to isolate the connection in the development environment from that in the production environment.

> **Note** Currently, only workspaces in standard mode support the connection isolation feature.

When you configure a sync node, the connection in the development environment is used. After you commit and deploy the sync node to the production environment for running, the connection in the production environment is used. To commit and deploy a node to the production environment for scheduling, you must configure a connection in both the development and production environments. The connection must have the same name in the development and production environments.

The connection isolation feature has the following impacts on workspaces:

- Workspaces in basic mode: The features and configuration dialog boxes of connections are the same as those before the connection isolation feature is added. For more information, see Connection configuration.
- Workspaces in standard mode: The Applicable Environment parameter is added to the configuration dialog boxes of connections.
- Workspaces upgraded from the basic mode to the standard mode: During the upgrade, you are prompted to upgrade connections. After the upgrade, the connections in the development environment are isolated from those in the production environment.

2.4.4.3. Sync data monitoring

The Sync Data Monitoring page displays the total number of sync node instances for different connections and the instance details based on the selected workspace and time range.

The cut-off time of data to be displayed is 0 minutes 0 seconds of the current hour. For example, if the current time is 2019-04-04 10:10:00, the page displays the data generated before 2019-04-04 10:00:00.

1. Log on to the DataWorks console and select a workspace.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, click **Sync Data Monitoring**. On the page that appears, view the total number of sync node instances for different connections and the instance details.
   - View summary data by connection type

     The Source and Target sections display the summary data of source connections and that of destination connections, respectively. Take source connections as an example. If the Source section displays MaxCompute with the value 1, a sync node instance whose source connection is MaxCompute is run in the selected time range.
   - View instance details

     The Sync Instances section displays the details of all sync node instances that are run in the selected time range. You can also perform the following operations:
     - Click a node in the **Node Name** column to go to the node configuration page.
     - Search for instances by condition, such as the ID, committer, node name, source connection type, and destination connection type. Sort search results based on the number of synchronized data entries or the size of synchronized data.

2.4.4.4. Manage connection permissions

DataWorks allows you to share connections among workspaces by managing permissions on the connections. After connections are shared, you can view the shared connections in the target workspaces. This topic describes how to manage permissions on connections and view shared connections.

Context

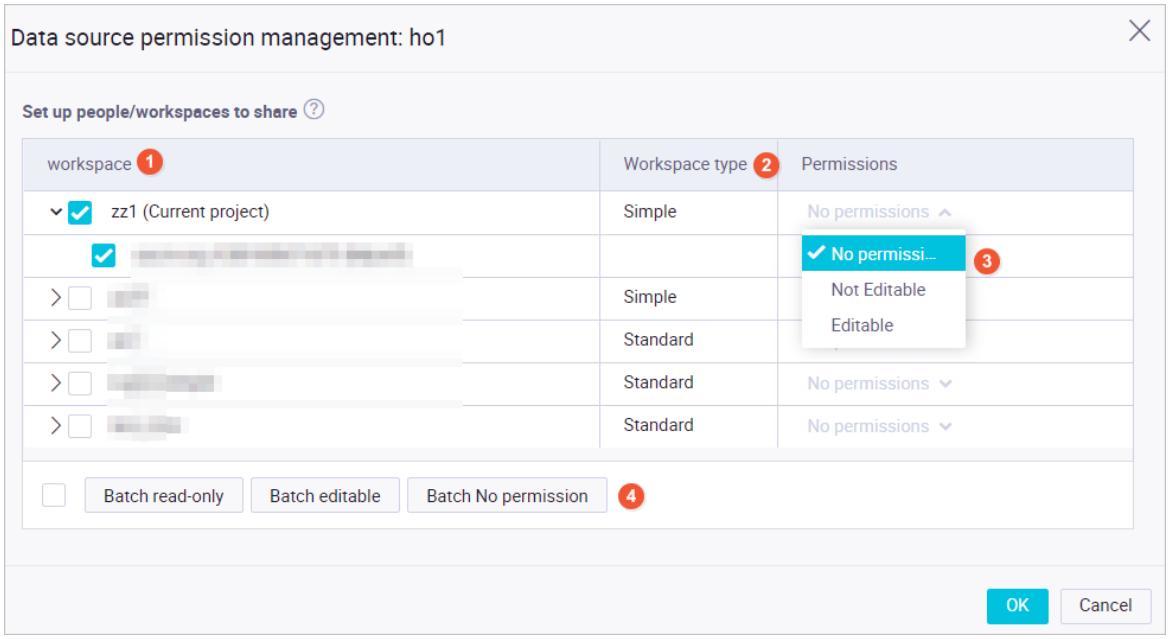
The configurations of a connection include sensitive information such as the endpoint of the data store, username, and password. Common developers only need to reference the connection to access the data store. Disclosing too much sensitive information or allowing

everyone to modify the configurations of the connection may cause security risks. If multiple users modify the configurations of a connection, the data store may fail to be connected. In this way, the nodes that reference the connection may fail.

Data Integration provides strict permission control. Only connection creators can manage the permissions on connections. They can grant permissions on connections to a specified workspace or user.

Go to the Data Source page

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Integration**.
3. On the page that appears, click **Data Source** in the left-side navigation pane.
4. On the page that appears, find the target connection and click **Modify Permission** in the **Actions** column.
5. In the **Data source permission management** dialog box, set the parameters as described in the following table.



No.	Parameter	Description

No.	Parameter	Description
1	**Workspace**	All workspaces that the current user joins and all members in each workspace. You can share the connection with several or all members in a workspace. ◦ If no permission is set for a connection, the connection inherits the permissions from the connection that is created earlier than the current one. ◦ When you configure the permissions on a connection for a workspace, the permissions apply to all members in the workspace. Members that join the workspace after the permission configuration also have the specified permissions. After you configure the permissions for a workspace, you can still configure the permissions for a specific user in the workspace. For example, after you set the permission on a connection to **No permission** for a workspace, you can still set the permission of a specific user in the workspace to **Editable**. ◦ You can configure the permissions on a connection for members in the current workspace. ◦ Only the creator of a connection can modify and share the connection. Other users including the workspace administrator cannot modify the connection. ◦ A workspace administrator can use a connection only after the workspace administrator is granted the required permission.
2	**Workspace type**	The type of each workspace. Valid values: **Simple** and **Standard**.
3	**Permissions**	The permission of a workspace or a member on the connection. Valid values: ◦ **No permission**: The workspace or member has no permission on the connection. ◦ **Not Editable**: The workspace or member can use the connection but cannot modify or view the configurations of connection. ◦ **Editable**: The workspace or member can use and modify the connection. **Note** If you grant the Editable permission with a workspace or member, the workspace or member can modify the connection. Exercise caution when you grant the Editable permission.
4	Batch operations	The operations that you can perform on the selected workspaces or members at a time. Valid values: **Batch read-only**, **Batch editable**, and **Batch No permission**.

6. Click **OK**.You can share connections across workspaces based on the following rules:
   - Between workspaces in simple mode:
     - If the source workspace is upgraded to the standard mode, connections in the production environment are shared.
     - If the target workspace is upgraded to the standard mode, a connection is shared to both the development environment and production environment with the same content.
   - From a workspace in simple mode to a workspace in standard mode: A connection is shared to both the development environment and production environment with the same content.
   - Between workspaces in standard mode: Connections in the development environment and production environment are shared to the corresponding environment separately.
   - From a workspace in standard mode to a workspace in simple mode:
     - You can share connections in both the production environment and development environment. Only connections in the production environment or development environment exist in the target workspace. If you share a connection in both environments, the newly shared one overrides the existing one in the target workspace.
     - If the target workspace is upgraded to the standard mode, the shared connection exists in both the development environment and production environment with the same content.

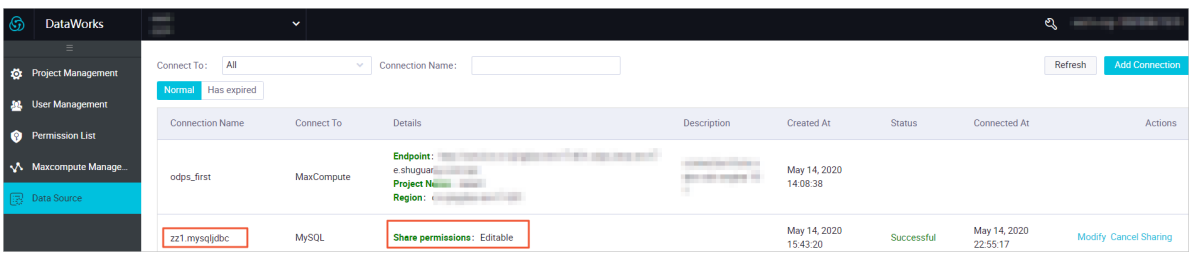
View shared connections

In the top navigation bar, select a workspace with connections shared from other workspaces from the drop-down list in the upper-left corner. The **Data Source** page of the selected workspace appears. On this page, you can view shared connections on the **Normal** and **Has expired** tabs.

- **Normal tab**

  On the Normal tab, you can view the information about each connection, including the connection name, connection type, permission details, connection description, creation time, connection status, and the time when the data store was last connected.

  The permission information appears in the **Details** column of the target connection. A shared connection is named in the Name of the workspace that shares the connection.Connection name format.

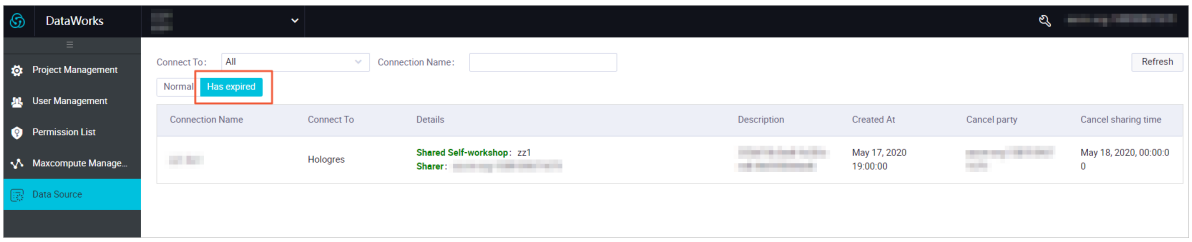
  

  If the current user has the Editable permission on the connection, **Modify** appears in the Actions column.

- **Has expired tab**

On the **Has expired** tab, you can view the connections for which your permissions have expired.

In the **Cancel party** column, you can view the member who revoked the permissions. In the **Created at** column, you can view the time when the permissions were revoked. The information helps you locate the cause of connection failures.



2.4.4.5. Configure a MySQL connection

A MySQL connection allows you to read data from and write data to MySQL by using MySQL Reader and Writer. You can configure sync nodes for MySQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **MySQL** in the Relational Databases section.
4. In the **Add MySQL Connection** dialog box, set the parameters as required.You can set Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a MySQL connection.
   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. Note This parameter is available only when the workspace is in standard mode.
RDS Instance ID	The ID of the ApsaraDB RDS for MySQL instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for MySQL instance.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

- The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. Note This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure a MySQL connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure MySQL Reader and Writer. For more information, see Configure the MySQL reader and Configure MySQL Writer.

2.4.4.6. Configure an SQL Server connection

An SQL Server connection allows you to read data from and write data to SQL Server by using SQL Server Reader and Writer. You can configure sync nodes for SQL Server by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **SQLServer** in the Relational Databases section.
4. In the **Add SQLServer Connection** dialog box, set the parameters as required.You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for an SQL Server connection
   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.

Parameter	Description
RDS Instance ID	The ID of the ApsaraDB RDS for SQL Server instance. You can view the ID in the ApsaraDB for RDS console.
RDS Instance Account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for SQL Server instance.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

- The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:sqlserver://ServerIP:Port;DatabaseName=Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure an SQL Server connection. You can proceed with the

next tutorial. In the next tutorial, you will learn how to configure SQL Server Reader and Writer. For more information, see Configure SQL Server Reader and Configure SQL Server Writer.

2.4.4.7. Configure a PostgreSQL connection

A PostgreSQL connection allows you to read data from and write data to PostgreSQL by using PostgreSQL Reader and Writer. You can configure sync nodes for PostgreSQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products** > **Data Aggregation** > **Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **PostgreSQL** in the Relational Databases section.
4. In the **Add PostgreSQL Connection** dialog box, set the required parameters.You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a PostgreSQL connection.
   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. Note This parameter is available only when the workspace is in standard mode.
RDS Instance ID	The ID of the ApsaraDB RDS for PostgreSQL instance. You can view the ID in the ApsaraDB for RDS console.

Parameter	Description
RDS Instance Account ID	The ID of the Apsara Stack tenant account that is used to purchase the ApsaraDB RDS for PostgreSQL instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

- The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:postgresql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.8. Configure an Oracle connection

An Oracle connection allows you to read data from and write data to Oracle by using Oracle Reader and Writer. You can configure sync nodes for Oracle by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Oracle** in the Relational Databases section.
4. In the **Add Oracle Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:oracle:thin:@ServerIP:Port:Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.9. Configure a Dameng connection

A Dameng connection allows you to read data from and write data to Dameng by using Dameng Reader and Writer. You can configure sync nodes for Dameng by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DM** in the Relational Databases section.
4. In the **Add DM Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:dm://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.10. Configure a DRDS connection

A DRDS connection allows you to read data from and write data to DRDS by using DRDS Reader and Writer. You can configure sync nodes for DRDS by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DRDS** in the Relational Databases section.
4. In the **Add DRDS Connection** dialog box, set the parameters as required.You can set the Connect To parameter to **ApsaraDB for DRDS** or **Connection Mode** for a DRDS connection.
   - The following table describes the parameters that appear after you set the Connect To parameter to **ApsaraDB for DRDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for DRDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the DRDS instance. You can view the ID in the DRDS console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the DRDS instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

   - The following table describes the parameters that appear after you set the Connect To parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

What's next

Now you have learned how to configure a DRDS connection. You can proceed with the next tutorial. In the next tutorial, you will learn how to configure DRDS Reader and Writer. For more information, see Configure the DRDS reader.

2.4.4.11. Configure a PolarDB connection

A PolarDB connection allows you to read data from and write data to PolarDB by using PolarDB Reader and Writer. You can configure sync nodes for PolarDB by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products** > **Data Aggregation** > **Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **POLARDB** in the Relational Databases section.

4. In the **Add POLARDB Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Database Type	The type of the database. Valid values: **MySql** and **Postgresql**.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.12. Configure a HybridDB for MySQL connection

A HybridDB for MySQL connection allows you to read data from and write data to HybridDB for MySQL by using HybridDB for MySQL Reader and Writer. You can configure sync nodes for HybridDB for MySQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HybridDB for MySQL** in the Relational Databases

section.

4. In the **Add HybridDB for MySQL Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for AnalyticDB**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the HybridDB for MySQL instance. You can view the ID in the HybridDB for MySQL console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the HybridDB for MySQL instance.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.13. Configure a HybridDB for PostgreSQL connection

A HybridDB for PostgreSQL connection allows you to read data from and write data to HybridDB for PostgreSQL by using HybridDB for PostgreSQL Reader and Writer. You can configure sync nodes for HybridDB for PostgreSQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products** > **Data Aggregation** > **Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HybridDB for PostgreSQL** in the Relational Databases section.

4. In the **Add HybridDB for PostgreSQL Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for AnalyticDB**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the HybridDB for PostgreSQL instance. You can view the ID in the HybridDB for PostgreSQL console.
Tenant Account ID	The ID of the Apsara Stack tenant account that is used to purchase the HybridDB for PostgreSQL instance. You can view your account ID on the **Security Settings** page.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.14. Configure an ApsaraDB for OceanBase connection

An ApsaraDB for OceanBase connection allows you to read data from and write data to ApsaraDB for OceanBase by using ApsaraDB for OceanBase Reader and Writer. You can configure sync nodes for ApsaraDB for OceanBase by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.

3. In the **Add Connection** dialog box, click **ApsaraDB for OceanBase** in the Big Data Storage Systems section.
4. In the **Add ApsaraDB for OceanBase Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the ApsaraDB for OceanBase database, in the format `jdbc:oceanbase://ip:port/database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.15. Configure a MaxCompute connection

A MaxCompute connection allows you to read data from and write data to MaxCompute by using MaxCompute Reader and Writer.

Context

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **MaxCompute** in the Big Data Storage Systems section.
4. In the **Add MaxCompute Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
ODPS Endpoint	The endpoint of the MaxCompute project. This parameter is read-only, and the value is automatically obtained from system configurations.
Tunnel Endpoint	The endpoint of the MaxCompute Tunnel service.
MaxCompute Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID for connecting to the MaxCompute project.
AccessKey Secret	The AccessKey secret for connecting to the MaxCompute project.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.16. Configure a DataHub connection

DataHub offers a comprehensive data import scheme to support fast computing for large amounts of data.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products** > **Data Aggregation** > **Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **DataHub** in the Big Data Storage Systems section.
4. In the **Add DataHub Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
DataHub Endpoint	The endpoint of DataHub. This parameter is read-only, and the value is automatically obtained from system configurations.
DataHub Project	The ID of the DataHub project.
AccessKey ID	The AccessKey ID for connecting to the DataHub project. You can view the AccessKey ID on the **User Info** page.
AceessKey Secret	The AccessKey secret for connecting to the DataHub project.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.17. Configure an AnalyticDB for MySQL connection

An AnalyticDB for MySQL connection allows you to write data to AnalyticDB for MySQL by using AnalyticDB for MySQL Writer. You can configure sync nodes for AnalyticDB for MySQL by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **ADS** in the Big Data Storage Systems section.
4. In the **Add ADS Connection** dialog box, set the parameters as required.

Parameter	Description

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Connection URL	The connection URL of AnalyticDB for MySQL, in the format of `Address:Port`.
Database	The name of the database.
AccessKey ID	The AccessKey ID for connecting to the AnalyticDB for MySQL database.
AccessKey Secret	The AccessKey secret for connecting to the AnalyticDB for MySQL database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.18. Configure a Vertica connection

A Vertica connection allows you to read data from and write data to Vertica by using Vertica Reader and Writer. You can configure sync nodes for Vertica by using the UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Vertica** in the Big Data Storage Systems section.
4. In the **Add Vertica Connection** dialog box, set the parameters as required.

Parameter	Description

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the Vertica database, in the format of `jdbc:vertica://Server IP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.19. Configure a GBase8a connection

A GBase8a connection allows you to read data from and write data to GBase8a by using GBase8a Reader and Writer. You can configure sync nodes for GBase8a by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **GBase8a** in the Big Data Storage Systems section.
4. In the **Add GBase8a Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
JDBC URL	The JDBC URL of the database, in the format of `jdbc:mysql://ServerIP:Port/Database`.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.20. Configure a Lightning connection

MaxCompute Lightning is an interactive query service that MaxCompute provides. MaxCompute Lightning complies with the PostgreSQL standards and syntax and allows you to use common tools and standard SQL to query and analyze data in MaxCompute projects.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Lightning** in the Big Data Storage Systems section.
4. In the **Add Lightning Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Host	The endpoint of the MaxCompute Lightning server. Default value: `seahawks.aliyun-inc.com`.
Port	The port number of the MaxCompute Lightning server. Default value: 8099.
Database Name	The name of the database.
Username and Password	The username and password that you can use to connect to the database.
ODPS Endpoint	The endpoint of MaxCompute.
MaxCompute Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID for connecting to the MaxCompute Lightning server.
AccessKey Secret	The AccessKey secret for connecting to the MaxCompute Lightning server.
JDBC Extension Parameters	The extension parameters used to establish a JDBC connection to MaxCompute Lightning. In this field, `prepareThreshold=0` is added by default and cannot be deleted. Otherwise, you cannot connect to MaxCompute Lightning.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.21. Configure an HBase connection

An HBase connection allows you to read data from and write data to HBase by using HBase Reader and Writer. You can configure sync nodes for HBase by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HBase** in the Big Data Storage Systems section.
4. In the **Add HBase Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.

Parameter	Description
Configuration	The HBase cluster configuration for client connections. You can convert the hbase-site.xml parameter to the JSON format and add more HBase client properties, such as cache and batch for scan operations, to optimize the interaction between the cluster and the client. Based on the edition of ApsaraDB for HBase in use, you must configure different information: ○ If you are using ApsaraDB for HBase Standard Edition or less advanced editions, the default configuration is used. You only need to enter the corresponding ZooKeeper information. ○ If you are using ApsaraDB for HBase editions that are more advanced than Standard Edition, the endpoint parameter specific to advanced editions is used for connection, and the zookeeper.quorum parameter is not used. The following configuration is an example for an HBase connection of ApsaraDB for HBase Enhanced Edition (Lindorm): `"hbaseConfig": {` `"hbase.client.connection.impl" : "com.alibaba.hbase.client.AliHBaseUEConnection",` `"hbase.client.endpoint" : "host:30020",` `"hbase.client.username" : "root",` `"hbase.client.password" : "root"` `}`

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.22. Configure a Hologres connection

A Hologres connection allows you to read data from and write data to Hologres by using Hologres Reader and Writer. You can configure sync nodes for Hologres by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.

ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**

iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Hologres** in the Big Data Storage Systems section.
4. In the **Add Hologres Connection** dialog box, set the parameters as required.

Parameter	Description
Connect To	The type of the connection. Default value: **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Instance ID	The ID of the Hologres instance.
Database Name	The name of the database in the Hologres instance.
AccessKey ID	The AccessKey ID for connecting to the Hologres database.
AccessKey Secret	The AccessKey secret for connecting to the Hologres database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.23. Configure a Hive connection

A Hive connection allows you to read data from and write data to Hive by using Hive Reader and Writer. You can configure sync nodes for Hive by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**

iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Hive** in the Big Data Storage Systems section.
4. In the **Add Hive Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
HIVE JDBC URL	The JDBC URL of the Hive metadatabase.
Database Name	The name of the Hive database. You can run the `show databases` command on the Hive client to query the created databases.
Logon Method	The mode for connecting to the Hive database. Valid values: **Username and password** and **Anonymous logon**. If you select **Username and password**, enter the username and password that you can use to connect to the Hive database.
metastoreUris	The URIs of the Hive metadatabase, in the format of `thrift://ip1:port1,thrift://ip2:por2`.
defaultFS	The address of the NameNode in the Active state in the HDFS, in the format of `hdfs://ip:port`.

Parameter	Description
Extension Parameters	The advanced parameters of Hive, such as those related to high availability (HA). The following code is an example: "hadoopConfig":{ "dfs.nameservices": "testDfs", "dfs.ha.namenodes.testDfs": "namenode1,namenode2", "dfs.namenode.rpc-address.youkuDfs.namenode1": "", "dfs.namenode.rpc-address.youkuDfs.namenode2": "", "dfs.client.failover.proxy.provider.testDfs "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider" }

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.24. Configure an OSS connection

Alibaba Cloud OSS is a secure and reliable service that allows you to store large amounts of objects.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **OSS** in the Semi-Structured Storage Systems section.
4. In the **Add OSS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.

Parameter	Description
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Endpoint	The OSS endpoint, in the format of `http://oss.aliyuncs.com`. The OSS endpoint varies with the region. **Note** If you add the bucket name before the domain name, for example, `http://xxx.oss.aliyuncs.com`, the connection can pass the connectivity test but data synchronization will fail.
Bucket	The name of the OSS bucket. A bucket is a storage space that serves as a container for storing objects. You can create one or more buckets and add one or more objects to each bucket. DataWorks can search for objects only in the bucket specified here during data synchronization.
AccessKey ID	The AccessKey ID for connecting to the OSS bucket.
AceessKey Secret	The AccessKey secret for connecting to the OSS bucket.

Notice When data in OSS is stored as CSV files, they must comply with the standard CSV format. For example, if the data in a column of a CSV file contains a double quotation mark ("), you must replace the double quotation mark with a pair of double quotation marks (""). Otherwise, the data in the CVS file may be incorrectly parsed.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.25. Configure an HDFS connection

A HDFS connection allows you to read data from and write data to HDFS by using HDFS Reader and Writer. You can configure sync nodes for HDFS by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.

ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**

iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **HDFS** in the Semi-Structured Storage Systems section.
4. In the **Add HDFS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
DefaultFS	The address of the NameNode in the HDFS, in the format of `hdfs://ServerIP:Port`.
Extension Parameters	The extension parameter hadoopConfig for HDFS Reader and Writer. You can configure the advanced parameters of Hadoop, such as those related to HA.

5. Click **Test Connection.**
6. After the connection passes the connectivity test, click **Complete.**

2.4.4.26. Configure an FTP connection

An FTP connection allows you to read data from and write data to FTP by using FTP Reader and Writer. You can configure sync nodes for FTP by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration.**
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **FTP** in the Semi-Structured Storage Systems section.
4. In the **Add FTP Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Portocol	The protocol used by the FTP server. Only FTP and SFTP are supported.
Host	The address of the FTP server.
Port	The port of the FTP server. The default port is 21 for FTP and 22 for SFTP.
Username	The username that you can use to connect to the FTP server.
Password	The password that you can use to connect to the FTP server.
Enable reverse VPC access	Specifies whether to enable reverse VPC access. Select the Enable check box if you cannot directly access the data store on an ECS instance but can access it by using a VPC.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.27. Configure a MongoDB connection

MongoDB is a document-oriented database that is second only to Oracle and MySQL. A MongoDB connection allows you to read data from and write data to MongoDB by using MongoDB Reader and Writer. You can configure sync nodes for MongoDB by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **MongoDB** in the NoSQL section.
4. In the **Add MongoDB Connection** dialog box, set the parameters as required.You can set the Connect To parameter to **ApsaraDB for RDS** or **Connection Mode** for a MongoDB connection.
   - **ApsaraDB for RDS**: Generally, the classic network is used to access the target ApsaraDB for MongoDB instance in this mode. You can access the ApsaraDB for MongoDB instance in the same region over the classic network. However, the access to the ApsaraDB for MongoDB instance from a different region over the classic network is not guaranteed to be successful.

Parameter	Description
Connect To	The connection type. In this example, set the value to **ApsaraDB for RDS**. **Note** If you have not assigned the default role to Data Integration, log on to the Resource Access Management (RAM) console with your Apsara Stack tenant account and perform authorization. Then, refresh this configuration page.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Region	The region where the ApsaraDB for MongoDB instance resides.
Instance ID	The ID of the ApsaraDB for MongoDB instance. You can view the ID in the ApsaraDB for MongoDB console.
Database Name	The name of the database that you created in the ApsaraDB for MongoDB console. You can also specify the database username and password in the console.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

- **Connection Mode**: Generally, the Internet is used to access the target database in this mode, which may cost you fees.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Address	The endpoint in the `host:port` format. To add an endpoint, click **Add Address** and specify the endpoint to add. To add more endpoints, repeat the preceding action. **Note** You must add either public endpoints or internal endpoints. Do not mix public endpoints with internal endpoints.
Database Name	The name of the database.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.28. Configure a Memcache connection

A Memcache connection allows you to write data to ApsaraDB for Memcache by using Memcache Writer. You can configure sync nodes for ApsaraDB for Memcache by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.

iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.

2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **Memcache(OCS)** in the NoSQL section.
4. In the **Add Memcache(OCS) Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Proxy Host	The IP address of the host or Memcache proxy. You can view the IP address on the basic information page of the ApsaraDB for Memcache console.
Port	The port for connecting to the ApsaraDB for Memcache instance. Default value: 11211.
Username	The username that you can use to connect to the database.
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.29. Configure a Redis connection

A Redis connection allows you to read data from and write data to Redis by using Redis Reader and Writer. You can configure sync nodes for Redis by using the code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click  in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.

3. In the **Add Connection** dialog box, click **Redis** in the **NoSQL** section.
4. In the **Add Redis Connection** dialog box, set the parameters as required.You can set the **Connect To** parameter to **ApsaraDB for RDS** or **Connection Mode** for a Redis connection.
   - The following table describes the parameters that appear after you set the **Connect To** parameter to **ApsaraDB for RDS**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **ApsaraDB for RDS**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Region	The region where the ApsaraDB for Redis instance resides.
Redis Instance ID	The ID of the ApsaraDB for Redis instance. You can view the ID in the ApsaraDB for Redis console.
Redis Password	The password that you can use to connect to the ApsaraDB for Redis instance. Leave it blank if no password is required.

   - The following table describes the parameters that appear after you set the **Connect To** parameter to **Connection Mode**.

Parameter	Description
Connect To	The type of the connection. In this example, set the value to **Connection Mode**.
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.

Parameter	Description
Server Address	The server address in the `host:port` format.
Add Server Address	Click **Add Server Address** to add a server address in the format of `host:port`.
Redis Password	The password that you can use to connect to Redis.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.30. Configure a Tablestore connection

Tablestore is a NoSQL database service built on Apsara distributed operating system. It allows you to store and access large amounts of structured data in real time.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click the icon in the upper-left corner and choose **All Products > Data Aggregation > Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **OTS** in the **NoSQL** section.
4. In the **Add OTS Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
Endpoint	The endpoint of the Tablestore service.
Table Store Instance ID	The name of the Tablestore instance.

Parameter	Description
AccessKey ID	The AccessKey ID for connecting to the Tablestore instance. You can view the AccessKey ID on the User Info page.
AccessKey Secret	The AccessKey secret for connecting to the Tablestore instance.

5. Click Test Connection.
6. After the connection passes the connectivity test, click Complete.

2.4.4.31. Configure an Elasticsearch connection

An Elasticsearch connection allows you to read data from and write data to Elasticsearch by using Elasticsearch Reader and Writer. You can configure sync nodes for Elasticsearch by using the code editor.

Procedure

1. Go to the Data Source page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose All Products > Data Aggregation > Data Integration.
   iii. On the page that appears, click Connection in the left-side navigation pane. The Data Source page appears.
2. On the Data Source page, click Add a Connection in the upper-right corner.
3. In the Add Connection dialog box, click ElasticSearch in the NoSQL section.
4. In the Add ElasticSearch Connection dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: Development and Production. Note This parameter is available only when the workspace is in standard mode.
Endpoint	The endpoint of Elasticsearch, in the format of `http://esxxxx.elasticsearch.aliyuncs.com:9200`.
Username	The username that you can use to connect to the database.

Parameter	Description
Password	The password that you can use to connect to the database.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.4.32. Configure a LogHub connection

A LogHub connection allows you to read data from and write data to LogHub by using LogHub Reader and Writer. You can configure sync nodes for LogHub by using the codeless UI or code editor.

Procedure

1. Go to the **Data Source** page.
   i. Log on to the DataWorks console.
   ii. On the DataStudio page, click in the upper-left corner and choose **All Products** > **Data Aggregation** > **Data Integration**.
   iii. On the page that appears, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
2. Click **Add Connection** in the upper-right corner.
3. In the **Add Connection** dialog box, click **LogHub** in the Message Queue section.
4. In the **Add LogHub Connection** dialog box, set the parameters as required.

Parameter	Description
Connection Name	The name of the connection. The name can contain letters, digits, and underscores (_). It must start with a letter.
Description	The description of the connection. The description can be up to 80 characters in length.
Applicable Environment	The environment in which the connection is used. Valid values: **Development** and **Production**. **Note** This parameter is available only when the workspace is in standard mode.
LogHub Endpoint	The LogHub endpoint, in the format of `http://cn-shanghai.log.aliyun.com`.
Project	The name of the LogHub project.
AccessKey ID	The AccessKey ID for connecting to the LogHub project. You can view the AccessKey ID on the **User Info** page.

Parameter	Description
AceessKey Secret	The AccessKey secret for connecting to the LogHub project.

5. Click **Test Connection**.
6. After the connection passes the connectivity test, click **Complete**.

2.4.5. Configure data synchronization tasks

2.4.5.1. Configure a sync node by using the codeless UI

This topic describes how to configure a sync node by using the codeless user interface (UI).

To configure a sync node, follow these steps:

1. Add connections.
2. Create a sync node.
3. Select a source connection.
4. Select a destination connection.
5. Map the fields in the source and destination tables.
6. Configure the channel, such as the maximum transmission rate and dirty data check rules.
7. Configure the node properties.

> **Note** The following sections describe the overall procedure. You can click the links in each step to read relevant instructions and then return to the current page to proceed with subsequent steps.

Add connections

Data synchronization is supported between various homogenous and heterogeneous connections. Before you configure a sync node, add required connections in Data Integration. Added connections are listed as options when you configure a sync node. For more information about connection types supported by Data Integration, see Supported data sources.

You can add connections of supported types to Data Integration. For more information about how to add a connection, see Data sources.

> **Note**
>
> - Data Integration does not support connectivity testing for some connection types. For more information, see Test data store connectivity.
> - Some connections are hosted on the premises. They do not have public IP addresses or network connections cannot be directly established. Such connections will fail the connectivity test. Data Integration allows you to add a custom resource group to resolve these issues. However, if you create sync nodes for such connections, you can only use the code editor. This is because you cannot obtain information such as table schema on the codeless UI if the network connection is unavailable.

Create a sync node

> Note This topic describes how to create and configure a sync node by using the codeless UI. Do not switch to the code editor.

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over the **Create** icon and select **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**. Then, click **Create**.
4. In the left-side navigation pane, click the created workflow. Then, right-click **Data Integration** and choose **Create Data Integration Node > Sync**. In the Create Node dialog box that appears, set **Node Name**.
5. Click **Commit**.

Select a source connection

After the sync node is created, configure the source connection and source table.

> Note
> - For more information about how to configure the source connection, see Configure the reader.
> - Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the parameter configuration feature of DataWorks to obtain the date and time required by incremental data synchronization.

Select a destination connection

After the source connection is configured, configure the destination connection and destination table.

> Note
> - For more information about how to configure the destination connection, see Configure the writer.
> - You can select the writing method for most nodes. For example, the writing method can be overwriting or appending. Supported writing methods vary with the connection type.

Map the fields in the source and destination tables

After the source and destination connections are configured, specify the mapping between the fields in the source and destination tables. You can click **Map Fields with the Same Name**, **Map Fields in the Same Line**, **Delete All Mappings**, and **Auto Layout**.

Button or icon	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	• Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. • You can use scheduling parameters, such as ${bizdate}. • You can enter functions supported by relational databases, such as now() and count(1). • Fields that cannot be parsed are indicated by Unidentified.

Note Make sure that the data type of a source field is the same as or compatible with that of the mapped destination field.

Configure channel control policies

When the preceding steps are completed, configure the channel control policies of the corresponding sync node.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure the node properties

This section describes how to use scheduling parameters for data filtering.

On the sync node configuration tab, click the **Properties** tab in the right-side navigation pane.

You can declare the scheduling parameters by using ${Variable name}. After a variable is declared, enter the initial value of the variable in the Arguments field. In this example, the initial value of the variable is identified by $[]. The content can be a time expression or a constant.

For example, if you write ${today} in the code and enter today=$[yyyymmdd] in the Arguments field, the value of the time variable is the current date. For more information about how to add and subtract the date, see Parameter configuration.

On the Properties tab, you can configure the properties of the sync node, such as the recurrence, scheduled time, and dependencies. Sync nodes have no ancestor nodes because their corresponding jobs are run before extract, transform, and load (ETL) jobs. We recommend that you specify the root node as their parent node.

Use custom scheduling parameters

To use custom scheduling parameters for the sync node, declare the following parameters in the code:

- bizdate: the timestamp of data to be used by the node. The value is one day before the running date of the node.
- cyctime: the time when the node is run, in the format of yyyymmddhhmiss.
- DataWorks provides the bizdate and cyctime parameters as default system parameters.

After the sync node is configured, save and commit the node.

2.4.5.2. Configure a sync node by using the code editor

This topic describes how to configure a sync node by using the code editor.

To configure a sync node, follow these steps:

1. Add connections.
2. Create a sync node.
3. Apply a template.
4. Configure the reader.
5. Configure the writer.
6. Map the fields in the source and destination tables.
7. Configure the channel, such as the maximum transmission rate and dirty data check rules.

8. Configure the node properties.

Add connections

Data synchronization is supported between various homogenous and heterogeneous connections. Before you configure a sync node, add required connections in Data Integration. Added connections are listed as options when you configure a sync node. For more information about connection types supported by Data Integration, see Supported data sources.

You can add connections of supported types to Data Integration. For more information about how to add a connection, see Data sources.

> **Note** Some connections are hosted on the premises. They do not have public IP addresses or network connections cannot be directly established. Such connections will fail the connectivity test. Data Integration allows you to add a custom resource group to resolve these issues. However, if you create sync nodes for such connections, you can only use the code editor. This is because you cannot obtain information such as table schema on the codeless user interface (UI) if the network connection is unavailable.

Create a sync node

> **Note** This topic describes how to create a sync node by using the codeless UI and configure the sync node by using the code editor.

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over the **Create** icon and select **Workflow**.
3. In the **Create Workflow** dialog box that appears, set **Workflow Name** and **Description**. Then, click **Create**.
4. In the left-side navigation pane, click the created workflow. Then, right-click **Data Integration** and choose **Create Data Integration Node > Sync**. In the Create Node dialog box that appears, set **Node Name**.
5. Click **Commit**.

Apply a template

1. After the sync node is created, the node configuration tab appears. Click the **Switch to Code Editor** icon in the toolbar.
2. In the **Confirm** dialog box that appears, click **OK** to switch to the code editor.

   > **Note** The code editor supports more features than the codeless UI. For example, you can configure sync nodes in the code editor even when the connectivity test fails.

3. Click the **Apply Template** icon in the toolbar.
4. In the **Apply Template** dialog box that appears, set **Source Connection Type**, **Connection**, **Target Connection Type**, and **Connection**.
5. Click **OK**.

Configure the reader

After the template is applied, the basic settings of the reader are configured. You can configure the source connection and source table as needed.

```
{"type": "job",
  "version": "2.0",
  "steps": [
    {
      "stepType": "mysql", // The reader type.
      "parameter": {
        "datasource": "MySQL", // The connection name.
        "column": [ // The columns to be synchronized.
          "id",
          "value",
          "table"
        ],
        "socketTimeout": 3600000, // The timeout period for reading data from and writing data to a s
ocket, in milliseconds.
        "connection": [
          {
            "datasource": "MySQL", // The connection name.
            "table": [
              "`case`" // The name of the table to be synchronized.
            ]
          }
        ],
        "where": "", // The WHERE clause.
        "splitPk": "", // The shard key.
        "encoding": "UTF-8" // The encoding format.
      },
      "name": "Reader",
      "category": "reader" // Indicates that these settings are related to the reader.
    },
```

The parameters are described as follows:

- type: the type of the sync node. You must set the value to job.
- version: the version number of the sync node. You can set the value to 1.0 or 2.0.

Note

- For more information about how to configure the source connection in the code editor, see Configure the reader.
- Incremental data synchronization is required when you configure the source connection for some sync nodes. In this case, you can use the parameter configuration feature of DataWorks to obtain the date and time required by incremental data synchronization.

Configure the writer

After the reader is configured, you can configure the destination connection and destination table as needed.

```
{
  "stepType": "odps", // The writer type.
  "parameter": {
     "partition": "", // The partitions that the reader reads.
     "truncate": true, // Specifies whether to clear up previous data and import new data when a write operation is performed again after failure. Set the value to true to guarantee the idempotence of write operations.
     "compress": false, // Specifies whether to enable compression.
     "datasource": "odps_first", // The connection name.
     "column": [ // The columns to be synchronized.
        "*"
     ],
     "emptyAsNull": false,
     "table": ""
    },
    "name": "Writer",
    "category": "writer" // Indicates that these settings are related to the writer.
  }
],
```

Note

- For more information about how to configure the destination connection in the code editor, see Configure the writer.
- You can select the writing method for most nodes. For example, the writing method can be overwriting or appending. Supported writing methods vary with the connection type.

Map the fields in the source and destination tables

The code editor only supports mapping of fields in the same row. Note that the data types of the fields must match.

> **Note** Make sure that the data type of a source field is the same as or compatible with that of the mapped destination field.

Configure channel control policies

When the preceding steps are completed, configure the channel control policies of the corresponding sync node. The setting parameter specifies the node efficiency, including the settings on the DUM number, thread concurrency, bandwidth throttling, dirty data policy, and resource group.

```
"setting": {
      "errorLimit": {
         "record": "1024" // The maximum number of dirty data records allowed.
      },
      "speed": {
         "throttle": false, // Specifies whether to enable bandwidth throttling.
         "concurrent": 1, // The maximum number of concurrent threads.
      }
   },
```

Setting	Description
Expected concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node in the code editor.
Bandwidth throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty data records allowed	The maximum number of dirty data records allowed.
Resource group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure the node properties

This section describes how to use scheduling parameters for data filtering.

On the sync node configuration tab, click the **Properties** tab in the right-side navigation pane.

On the Properties tab, you can configure the properties of the sync node, such as the recurrence, scheduled time, and dependencies. Sync nodes have no ancestor nodes because their corresponding jobs are run before extract, transform, and load (ETL) jobs. We recommend that you specify the root node as their parent node.

After the sync node is configured, save and commit the node.

2.4.5.3. Configure the reader

2.4.5.3.1. Configure DRDS Reader

Distributed Relational Database Service (DRDS) Reader allows you to read data from DRDS. DRDS Reader connects to a remote DRDS database and runs a SELECT statement to select and read data from the database.

Currently, DRDS Reader only supports MySQL engines. DRDS is a distributed MySQL database service that complies with MySQL protocols in most cases.

Specifically, DRDS Reader connects to a remote DRDS database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The DRDS database runs the statement and returns the result. Then, DRDS Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

DRDS Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the DRDS database. DRDS does not support all MySQL specifications, such as JOIN statements.

DRDS Reader supports most DRDS data types. Make sure that your data types are supported.

The following table lists the data types supported by DRDS Reader.

Category	DRDS data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
table	The name of the table to be synchronized.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id", "`table`", "1", "'bazhen.csy'", "null", "to_char(a + 1)", "2.3", "true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ bazhen.csy: a string constant. ○ null: a null pointer. ○ to_char(a + 1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None
where	The WHERE clause. DRDS Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `STRTODATE('${bdp.system.bizdate}', '%Y%m%d') <= today AND today < DATEADD(STRTODATE('${bdp.system.bizdate}', '%Y%m%d'), interval 1 day)`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None

Configure DRDS Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. Note The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.

Parameter	Description
Add	Click Add to add a field. The rules for adding fields are described as follows: ○ You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless user interface (UI).
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure DRDS Reader by using the code editor

In the following code, a node is configured to read data from a DRDS database.

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"drds", // The reader type.
      "parameter":{
        "datasource":"", // The connection name.
        "column":[ // The columns to be synchronized.
          "id",
```

```
                "name"
            ],
            "where":"", // The WHERE clause.
            "table":"", // The name of the table to be synchronized.
            "splitPk": "" // The shard key.
        },
        "name":"Reader",
        "category":"reader"
      },
      {// The following template is used to configure Stream Writer. For more information about how to configure other writers, see the corresponding topic.
        "stepType":"stream", // The writer type.
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0" // The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false, // Specifies whether to enable bandwidth throttling.
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}:"Writer"
        }
      ]
    }
}
```

Additional instructions

- Consistency

  As a distributed database service, DRDS cannot provide a consistent view of multiple tables in multiple databases. Different from MySQL where data is synchronized in a single table of a single database, DRDS Reader cannot extract the snapshot of database and table shards at the same time slice. That is, DRDS Reader extracts different snapshots from different shards. As a result, this cannot guarantee strong consistency for data queries.

- Character encoding

  DRDS supports flexible encoding configurations. You can specify the encoding format for an instance, a field, a table, and a database. The configurations for the field, table, database, and instance are prioritized in descending order. We recommend that you use UTF-8 for a database.

  DRDS Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

  If you specify the encoding format for a DRDS database but data is written to the DRDS database in a different encoding format, DRDS Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization

  DRDS Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, DRDS Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  DRDS Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.4.5.3.2. Configure HBase Reader

HBase Reader allows you to read data from HBase. HBase Reader connects to a remote HBase database through a Java client of HBase. Then, HBase Reader scans and reads data based on the specified rowkey range, assembles the data to abstract datasets in custom data types supported by Data Integration, and then passes the datasets to a writer.

Data types

The following table lists the data types supported by HBase Reader.

Category	Data Integration data type	HBase data type
Integer	LONG	Short, Int, and Long
Floating point	DOUBLE	Float and Double
String	STRING	Binary_String and String
Date and time	DATE	Date
Byte	BYTES	Bytes
Boolean	BOOLEAN	Boolean

Parameters

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required. Note • If the value is true, the following five Kerberos-related parameters must be specified: ◦ kerberosKeytabFilePath ◦ kerberosPrincipal ◦ hbaseMasterKerberosPrincipal ◦ hbaseRegionserverKerberosPrincipal ◦ hbaseRpcProtection • If the value is false, Kerberos authentication is not required and you do not need to specify the preceding parameters.	No	*false*
hbaseConfig	The properties of the HBase cluster, in JSON format. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. You can also configure other properties, such as those related to the cache and batch for scan operations.	Yes	None
mode	The mode in which data is read from the HBase connection. Valid values: normal and multiVersionFixedColumn.	Yes	None
table	The name of the HBase table from which data is read. The name is case-sensitive.	Yes	None

Parameter	Description	Required	Default value
encoding	The encoding format, by using which binary data stored in byte[] format is converted into strings. Currently, UTF-8 and GBK are supported.	No	UTF-8
column	The HBase columns from which data is read. • In normal mode: The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey. The type parameter specifies the source data type. The format parameter specifies the date format. The value parameter specifies the column value if the column is a constant column. Example: "column": [{ "name": "rowkey", "type": "string" }, { "value": "test", "type": "string" }] For the column parameter, you must specify the type parameter and specify one of the name and value parameters. • In multiVersionFixedColumn mode: The name parameter specifies the name of the column in the HBase table. The format must be columnFamily:columnName except for the rowkey. The type parameter specifies the source data type. The format parameter specifies the date format. You cannot create constant columns in multiVersionFixedColumn mode. Example:	Yes	None

Parameter	Description	Required	Default value
	"column": [{ "name": "rowkey", "type": "string" }, { "name": "info:age", "type": "string" }]		
maxVersion	The number of versions read by HBase Reader when multiple versions are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions are read.	Required in multiVersionFixedColumn mode	None
range	The rowkey range that HBase Reader reads. • startRowkey: the start rowkey. • endRowkey: the end rowkey. • isBinaryRowkey: the method used to convert the specified start and end rowkeys into the byte[] format. Default value: false. If the value is true, Bytes.toBytesBinary(rowkey) is used. If the value is false, Bytes.toBytes(rowkey) is used. Example: "range": { "startRowkey": "aaa", "endRowkey": "ccc", "isBinaryRowkey":false }	No	None
scanCacheSize	The number of rows read by an HBase client with each remote procedure call (RPC) connection.	No	*256*
scanBatchSize	The number of columns read by an HBase client with each RPC connection.	No	*100*

Configure HBase Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HBase Reader.

Configure HBase Reader by using the code editor

In the following code, a node is configured to read data from an HBase connection in normal mode.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "hbase", // The reader type.
      "parameter": {
        "mode": "normal",
        "scanCacheSize": 256, // The number of rows read by an HBase client with each RPC connecti
on.
        "scanBatchSize": 256, // The number of columns read by an HBase client with each RPC conne
ction.
        "hbaseVersion": "094x",
        "datasource": "demo_hbase", // The connection name.
        "column": [
          {
            "name": "info:idx",
            "type": "long"
          },
          {
            "name": "info:age",
            "type": "string"
          },
          {
            "name": "info:birthday",
            "format": "yyyy-MM-dd",
            "type": "date"
          }
        ],
        "range": {
          "startRowKey": "", // The start rowkey.
          "endRowKey": "", // The end rowkey.
          "isBinaryRowKey": false // The method used to convert the specified start and end rowkey
s into the byte[] format. Default value: false. If the value is true, Bytes.toBytesBinary(rowkey) is used.
If the value is false, Bytes.toBytes(rowkey) is used.
        },
        "maxVersion": , // The number of versions read by HBase Reader when multiple versions are
```

```
available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions are rea
d.
            "encoding": "UTF-8",
            "table": "test" // The name of the HBase table from which data is read. The name is case-sen
sitive.
        },
        "name": "Reader",
        "category": "reader"
      },
        "tepType": "odps", // The writer type.
        "parameter": {},
        "name": "Writer",
        "category": "writer"
      }
    ],
    "setting": {
    },
    "order": {
      "hops": [
        {
          "from": "Reader",
          "to": "Writer"
        }
      ]
    }
}
```

In the following code, a node is configured to read data from an HBase connection in multiVersionFixedColumn mode.

```
{
    "type": "job",
    "version": "2.0", // The version number.
    "steps": [
      {
        "stepType": "hbase", // The reader type.
        "parameter": {
              "table": "users", // The name of the HBase table from which data is read. The name is case-
sensitive.
              "encoding": "utf-8", // The encoding format, by using which binary data stored in byte[] for
mat is converted into strings. Currently, UTF-8 and GBK are supported.
```

```
            "mode": "multiVersionFixedColumn",
            "maxVersion": "-1", // The number of versions read by HBase Reader when multiple version
s are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions ar
e read.
            "column": [ // The HBase columns from which data is read. The name parameter specifies th
e name of the column in the HBase table. The format must be columnFamily:columnName except for th
e rowkey. The type parameter specifies the source data type. The format parameter specifies the dat
e format. You cannot create constant columns in multiVersionFixedColumn mode.
              {
                "name": "rowkey",
                "type": "string"
              },
              {
                "name": "info: age",
                "type": "string"
              },
              {
                "name": "info: birthday",
                "type": "date",
                "format": "yyyy-MM-dd"
              }
            ],
            "range": { // The rowkey range that HBase Reader reads.
              "startRowkey": "",
              "endRowkey": ""
            }
          }
        },
        "name": "Reader",
        "category": "reader"
      },
      {
        "stepType": "odps", // The writer type.
        "parameter": {},
        "name": "Writer",
        "category": "writer"
      }
    ],
    "setting": {
    },
    "order": {
```

```
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  }
}
```

2.4.5.3.3. Configure HDFS Reader

HDFS Reader allows you to read data stored in a Hadoop Distributed File System (HDFS). HDFS Reader connects to an HDFS, reads data from files in the HDFS, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

Examples:

TextFile is the default storage format for creating Hive tables, without data compression. Essentially, a TextFile file is stored in HDFS as text. For Data Integration, the implementation of HDFS Reader is similar to that of OSS Reader.

Optimized Row Columnar File (ORCFile) is an optimized RCFile format. It provides an efficient method for storing Hive data. HDFS Reader uses the OrcSerde class provided by Hive to read and parse ORCFile data.

> Note
>
> - Considering that a complex network connection is required between the default resource group and HDFS, we recommend that you use a custom resource group to run sync nodes. Make sure that your custom resource group can access the NameNode and DataNode of HDFS through a network.
> - By default, HDFS uses a network whitelist to guarantee data security. In this case, we recommend that you use a custom resource group to run HDFS sync nodes.
> - If you configure an HDFS sync node in the code editor, the HDFS connection does not need to pass the connectivity test. In this case, you can temporarily ignore connectivity test errors.
> - To synchronize data in Data Integration, you must log on as an administrator. Make sure that you have the permissions to read data from and write data to relevant HDFS files.

Features

Currently, HDFS Reader supports the following features:

- Supports the TextFile, ORCFile, RCFile, SequenceFile, CSV, and Parquet file formats. What is stored in each file must be a logical two-dimensional table.
- Reads data of various types as strings. Supports constants and column pruning.
- Supports recursive reading. Supports regular expressions that contain asterisks (*) and question marks (?).

- Compresses ORCFile files in SNAPPY or ZLIB format.
- Compresses SequenceFile files in LZO format.
- Reads multiple files concurrently.
- Compresses CSV files in GZIP, BZIP2, ZIP, LZO, LZO_DEFLATE, or SNAPPY format.
- Supports Hive 1.1.1 and Hadoop 2.7.1 (compatible with Apache JDK 1.6). HDFS Reader can work properly with Hadoop 2.5.0, Hadoop 2.6.0, and Hive 1.2.0 during testing.

> **Note** Currently, HDFS Reader cannot use concurrent threads to read a single file.

Data types

RCFile

RCFile metadata is stored in databases managed by Hive, and in different formats depending on the data type. However, HDFS Reader cannot query metadata from such databases. If you want to synchronize a file of the RCFile format, you must specify the data type for each column. If the data type is BIGINT, DOUBLE, or FLOAT, specify the data type as BIGINT, DOUBLE, or FLOAT. If the data type is VARCHAR or CHAR, specify the data type as STRING.

RCFile data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	STRING, CHAR, and VARCHAR
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN
Binary	BINARY

Parquet files

Parquet file data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	INT32, INT64, and INT96
Floating point	FLOAT and DOUBLE
String	FIXED_LEN_BYTE_ARRAY
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN

Category	HDFS data type
Binary	BINARY

TextFile, ORCFile, and SequenceFile

TextFile metadata and ORCFile metadata are stored in databases, such as MySQL databases, managed by Hive. However, HDFS Reader cannot query metadata from such databases. If you want to convert data types during data synchronization, you must specify the data types.

TextFile, ORCFile, and SequenceFile data types are automatically converted into the data types supported by Data Integration. The following table lists the supported data types.

Category	HDFS data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT and DOUBLE
String	STRING, CHAR, VARCHAR, STRUCT, MAP, ARRAY, UNION, and BINARY
Date and time	DATE and TIMESTAMP
Boolean	BOOLEAN

The data types are described as follows:

- LONG: integer strings in HDFS files, such as 123456789.
- DOUBLE: double value strings in HDFS files, such as 3.1415.
- BOOLEAN: Boolean strings in HDFS files, such as true and false. The strings are case-insensitive.
- DATE: date and time strings in HDFS files, such as 2014-12-31 00:00:00.

Note The TIMESTAMP data type of Hive is accurate to nanoseconds. If you convert TIMESTAMP-type Hive data, such as 2015-08-21 22:40:47.397898389, in TextFile and ORCFile files into the DATE type in Data Integration, the converted data is accurate to seconds. If you need nanosecond-scale accuracy, convert TIMESTAMP-type data into the STRING type in Data Integration.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
path	The path of the file to read. To read multiple files, use a regular expression such as /hadoop/data_201704*. • If you specify a single HDFS file, HDFS Reader uses only one thread to read the file. • If you specify multiple HDFS files, HDFS Reader uses multiple threads. The number of threads is limited by the transmission rate, in Mbit/s. Note The actual number of threads is determined by both the number of HDFS files to be read and the specified transmission rate. • When a path contains a wildcard, HDFS Reader attempts to read all files that match the path. If the path is ended with a slash (/), HDFS Reader reads all files in the specified directory. For example, if you specify the path as /bazhen/, HDFS Reader reads all files in the bazhen directory. Currently, HDFS Reader only supports asterisks (*) and question marks (?) as file name wildcards. The syntax is similar to that of file name wildcards used on the Linux command line. Note • Data Integration considers all the files on a sync node as a single table. Make sure that all the files on each sync node can adapt to the same schema and Data Integration has the permission to read all these files. • Note: When creating Hive tables, you can specify partitions. For example, if you specify partition(day="20150820",hour="09"), a directory named /20150820 and a subdirectory named /09 are created in the corresponding table directory of the HDFS. Therefore, if you need HDFS Reader to read the data of a partition, specify the file path of the partition. For example, if you need HDFS Reader to read all the data in the partition with the date of 20150820 in the table named mytable01, specify the path as follows: "path": "/user/hive/warehouse/mytable01/20150820/*"	Yes	None

Parameter	Description	Required	Default value
defaultFS	The address of the NameNode of the HDFS. If a sync node is run on the default resource group, advanced parameter settings of Hadoop, such as those related to high availability, are not supported.	Yes	None

Parameter	Description	Required	Default value
fileType	The file format. Valid values: text, orc, rc, seq, csv, and parquet. HDFS Reader automatically recognizes the file format and uses corresponding read policies. Before data synchronization, HDFS Reader checks whether all the source files match the specified format. If any source file does not match the format, the sync node fails. The valid values of the fileType parameter are described as follows: • text: the TextFile format. • orc: the ORCFile format. • rc: the RCFile format. • seq: the SequenceFile format. • csv: the common HDFS file format, that is, the logical two-dimensional table. • parquet: the common Parquet file format. Note TextFile and ORCFile are different formats. HDFS Reader parses files in the two formats in different ways. After being converted from a composite data type of Hive into the STRING type of Data Integration, the data in a file of the TextFile format can be different from that in the same file of the ORCFile format. Composite data types include MAP, ARRAY, STRUCT, and UNION. The following example uses the conversion from the MAP type to the STRING type as an example: • HDFS Reader converts MAP-type ORCFile data into a string: {job=80, team=60, person=70}. • HDFS Reader converts MAP-type TextFile data into a string: job:80, team:60, person:70. The conversion results show that the data remains unchanged but the formats differ slightly. Therefore, if the data to be synchronized matches a composite data type of Hive, we recommend that you use a uniform file format. Recommendations: • To use a uniform file format, we recommend that you export TextFile tables as ORCFile tables on the Hive client. • If the file format is Parquet, the parquetSchema parameter is required, which specifies the schema of the Parquet table. For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, HDFS Reader reads all data as strings. Specify this parameter as `"column":["*"]`. You can also specify the column parameter in the following way: { "type": "long", "index": 0 // The first INT-type column of the source file. }, { "type": "string", "value": "alibaba" // The value of the current column, that is, a constant "alibaba". }	Yes	None
fieldDelimiter	The column delimiter. To read TextFile data, you must specify the column delimiter. The default delimiter is comma (,). To read ORCFile data, you do not need to specify the column delimiter. The default delimiter is \u0001. • If you need each row to be converted into a column in the destination table, use a string that does not exist in every row, such as \u0001. • Do not use \n as the delimiter.	No	,
encoding	The encoding format of the file to read.	No	UTF-8
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify nullFormat:"null", Data Integration considers null as a null pointer.	No	None

Parameter	Description	Required	Default value
compress	The compression format. Available compression formats for CSV files are GZIP, BZIP2, ZIP, LZO, LZO_DEFLATE, and SNAPPY. Note • Do not mix up LZO with LZO_DEFLATE. • Snappy does not have a uniform stream format. Data Integration currently only supports the most popular two compression formats: hadoop-snappy (Snappy stream format in Hadoop) and framing-snappy (Snappy stream format recommended by Google). • rc indicates the RCFile format. • This parameter is not required for files of the ORCFile format.	No	None

Parameter	Description	Required	Default value
parquetSchema	The schema of the source file. This parameter is required only when the fileType parameter is set to parquet. Format: `message messageTypeName {` `required, dataType, columnName;` `..... ;` `}` The format is described as follows: • messageTypeName: the name of the MessageType object. • required: specifies whether the field is required or optional. We recommend that you set the parameter to optional for all fields. • dataType: the data type of the field. Supported data types: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY. Select BINARY if the data type is STRING. Note Each line, including the last one, must end with a semicolon (;). An example is provided as follows: `message m {` `optional int64 id;` `optional int64 date_id;` `optional binary datetimestring;` `optional int32 dspId;` `optional int32 advertiserId;` `optional int32 status;` `optional int64 bidding_req_num;` `optional int64 imp;` `optional int64 click_num;` `}`	No	None

Parameter	Description	Required	Default value
csvReaderConfig	The configurations for reading CSV files. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV files, which supports many configurations. The following example provides common configurations: `"csvReaderConfig":{` `"safetySwitch": false,` `"skipEmptyRecords": false,` `"useTextQualifier": false` `}` You can use the following parameters and their default values: `boolean caseSensitive = true;` `char textQualifier = 34;` `boolean trimWhitespace = true;` `boolean useTextQualifier = true; // Specifies whether to use escape characters for CSV files.` `char delimiter = 44; // The delimiter.` `char recordDelimiter = 0;` `char comment = 35;` `boolean useComments = false;` `int escapeMode = 1;` `boolean safetySwitch = true; // Specifies whether to limit the length of each column to 100,000 characters.` `boolean skipEmptyRecords = true; // Specifies whether to skip empty rows.` `boolean captureRawRecord = true;`	No	None

Parameter	Description	Required	Default value
hadoopConfig	The advanced parameter settings of Hadoop, such as those related to high availability. "hadoopConfig":{ "dfs.nameservices": "testDfs", "dfs.ha.namenodes.testDfs": "namenode1,namenode2", "dfs.namenode.rpc-address.youkuDfs.namenode1": "", "dfs.namenode.rpc-address.youkuDfs.namenode2": "", "dfs.client.failover.proxy.provider.testDfs": "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider" }	No	None

Configure HDFS Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HDFS Reader.

Configure HDFS Reader by using the code editor

In the following code, a node is configured to read data from an HDFS. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "version": "2.0",
  "steps": [
    {
      "stepType": "hdfs", // The reader type.
      "parameter": {
        "path": "", // The path of the file to read.
        "datasource": "", // The connection name.
        "column": [
          {
            "index": 0, // The ID of the column in the source table.
            "type": "string" // The data type.
          },
          {
            "index": 1,
            "type": "long"
          },
          {
```

```
                "index": 2,
                "type": "double"
            },
            {
                "index": 3,
                "type": "boolean"
            },
            {
                "format": "yyyy-MM-dd HH:mm:ss", // The format of the time.
                "index": 4,
                "type": "date"
            }
        ],
        "fieldDelimiter": "," // The column delimiter.
        "encoding": "UTF-8", // The encoding format.
        "fileType": "" // The file format.
      },
      "name": "Reader",
      "category": "reader"
    },
    {// The following template is used to configure the writer. For more information, see the corresponding topic.
      "stepType": "stream",
      "parameter": {},
      "name": "Writer",
      "category": "writer"
    }
  ],
  "setting": {
    "errorLimit": {
      "record": "" // The maximum number of dirty data records allowed.
    },
    "speed": {
      "concurrent": 3, // The maximum number of concurrent threads.
      "throttle": false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
    }
  },
  "order": {
    "hops": [
```

```
        "hops": [
          {
            "from": "Reader",
            "to": "Writer"
          }
        ]
      }
    }
```

2.4.5.3.4. Configure MaxCompute Reader

This topic describes the data types and parameters supported by MaxCompute Reader and how to configure it by using the codeless UI and code editor.

MaxCompute Reader can read data from a MaxCompute project by using the MaxCompute Tunnel service based on the source project, table, partition, and table fields that you have configured.

MaxCompute Reader cannot read views. It can read only partitioned tables and non-partitioned tables. To allow MaxCompute Reader to read partitioned tables, you must specify the partition information. For example, set pt to 1 and ds to hangzhou for the t0 table. The partition information is not required for non-partitioned tables. Additionally, you can select some or all of the table fields, change the order in which the fields are arranged, or add constant fields and partition key columns. Note that partition key columns are not table fields.

Data types

The following table lists the data types supported by MaxCompute Reader.

Category	Data Integration data type	MaxCompute data type
Integer	LONG	BIGINT, INT, TINYINT, and SMALLINT
Boolean	BOOLEAN	BOOLEAN
Date and time	DATE	DATETIME and TIMESTAMP
Floating point	DOUBLE	FLOAT, DOUBLE, and DECIMAL
Binary	BYTES	BINARY
Complex	STRING	ARRAY, MAP, and STRUCT

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the source table. The name is not case-sensitive.	Yes	None
partition	The partitions that MaxCompute Reader reads. Linux shell wildcards are supported. An asterisk (*) represents zero or more characters, and a question mark (?) represents that the previous character can be included or not. Assume that a partitioned table named test has four partitions: pt=1 and ds=hangzhou, pt=1 and ds=shanghai, pt=2 and ds=hangzhou, and pt=2 and ds=beijing. • To read data from the partition with pt=1 and ds=shanghai, enter `"partition":"pt=1/ds=shanghai"`. • To read data from all the partitions with pt=1, enter `"partition":"pt=1/ds=*"`. • To read data from all the partitions in the test table, enter `"partition":"pt=*/ds=*"`.	Required only for writing data to a partitioned table	None

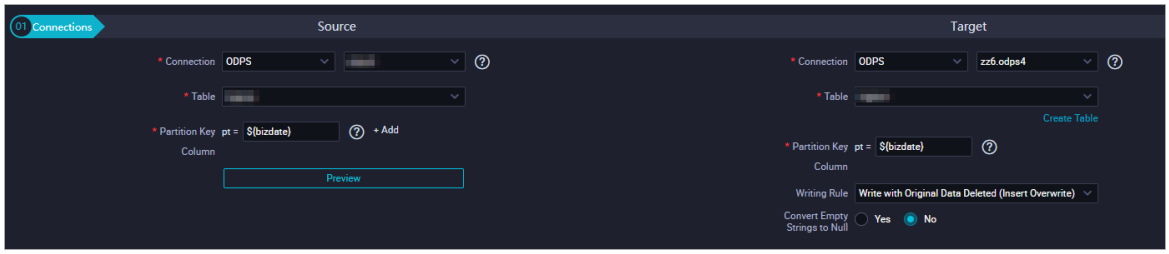
Parameter	Description	Required	Default value
column	The columns in the source table that MaxCompute Reader reads. Assume that the fields of a table named test are id, name, and age. • To read the fields in turn, enter `"column":["id","name","age"]` or `"column":["*"]`. Note We recommend that you do not set "column":["*"]. This is because data synchronization may fail if the source table changes in the column order, data type, or number of columns. • To read the name and id fields in turn, enter `"column":["name","id"]`. • You can add a constant field to extracted data for the purpose of proper mapping between source table columns and destination table columns. Each constant must be enclosed in single quotation marks (`''`). For example, if you set `"column":["age","name","'1988-08-08 08:08:08'","id"]`, the data extracted contains an age column, a name column, a constant "1988-08-08 08:08:08", and an id column in turn. The single quotation marks (`''`) are used to identify constant columns. The constant column values exclude the single quotation marks (`''`). Note ◦ MaxCompute Reader does not use SELECT statements to read data. Therefore, you cannot specify function fields. ◦ The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Configure MaxCompute Reader by using the codeless UI

On the **DataStudio** page, create a sync node under a workflow and configure the node.

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Partition Key Column	The partition information. You can click **Add** on the right to add partition key columns.

Note To synchronize all columns in the source table, enter "column":[""]. The partition parameter supports wildcards and includes one or more partitions.

- `"partition":"pt=20140501/ds=*"` specifies that all ds partitions with pt=20140501 are to be synchronized.
- `"partition":"pt=top?"` specifies that the partitions with pt=top and pt=to are to be synchronized.

You can specify the partition key columns to be synchronized, such as a partition key column named pt. Assume that the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. You can configure the column to be synchronized to pt. Ignore it if the column is marked as unidentified. To synchronize all partitions, enter pt=*. To synchronize specified partitions, specify the corresponding dates.

2. Configure field mapping. It is equivalent to setting the column parameter provided in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.

GUI element	Description
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MaxCompute Reader by using the code editor

In the following code, a node is configured to read data from MaxCompute. For more information about the parameters, see the preceding parameter description.

```
{
   "type":"job", // The type of the sync node.
   "version":"2.0", // The version number.
   "steps":[
```

```
    {
      "stepType":"odps",// The reader type.
      "parameter":{
        "partition":[], // The partitions that MaxCompute Reader reads.
        "isCompress":false, // Specifies whether to enable compression.
        "datasource":"",// The connection name.
        "column":[// The columns to be synchronized.
          "id"
        ],
        "emptyAsNull":true,
        "table":""// The table name.
      },
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"stream", // The writer type.
      "parameter":{ // The parameters that you specify for the writer.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0" // The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader", // The source connection of the node.
        "to":"Writer" // The destination connection of the node.
      }
```

```
    ]
  }
}
```

2.4.5.3.5. Configure MongoDB Reader

This topic describes the data types and parameters supported by MongoDB Reader and how to configure it by using the code editor.

MongoDB Reader connects to a remote MongoDB database by using the Java client named MongoClient and reads data from the database. The latest version of MongoDB has improved the locking feature from database locks to document locks. By using the powerful functionalities of indexes in MongoDB, MongoDB Reader can efficiently read data from MongoDB databases.

> Note
>
> - If you use ApsaraDB for MongoDB, the MongoDB database has a root account by default. For security concerns, Data Integration supports access to a MongoDB database only by using a MongoDB database account. When you add a MongoDB connection, do not use the root account for access.
> - JavaScript syntax is not supported for queries.

MongoDB Reader shards data in the MongoDB database based on specified rules, reads data from the database with multiple threads, and then converts the data to a format readable by Data Integration.

Data types

MongoDB Reader supports most MongoDB data types. Make sure that your data types are supported.

The following table lists the data types supported by MongoDB Reader.

Category	MongoDB data type
Long	INT, LONG, DOCUMENT.INT, and DOCUMENT.LONG
Double	DOUBLE and DOCUMENT.DOUBLE
String	STRING, ARRAY, DOCUMENT.STRING, DOCUMENT.ARRAY, and COMBINE
Date	DATE and DOCUMENT.DATE
Boolean	BOOLEAN and DOCUMENT.BOOLEAN
Bytes	BYTES and DOCUMENT.BYTES

Note

- The DOCUMENT data type is used to store embedded documents. It is also called the OBJECT data type.
- The following content describes how to use the COMBINE data type:

  When MongoDB Reader reads data from a MongoDB database, it combines and converts multiple fields in MongoDB documents to a JSON string.

  For example, doc1, doc2, and doc3 are three MongoDB documents with different fields, which are represented by keys instead of key-value pairs. The keys a and b represent common fields in all the three documents. The key x_n represents an unfixed field.

  ```
  doc1: a b x_1 x_2
  ```

  ```
  doc2: a b x_2 x_3 x_4
  ```

  ```
  doc3: a b x_5
  ```

  To import the preceding three MongoDB documents to MaxCompute, you must specify the fields to retain, set a name for each combined string, and set the data type of each combined string to COMBINE in the configuration file. Make sure that the name of each combined string is unique among all existing fields in the documents.

  ```
  "column": [
  {
  "name": "a",
  "type": "string",
  },
  {
  "name": "b",
  "type": "string",
  },
  {
  "name": "doc",
  "type": "combine",
  }
  ]
  ```

  The following table lists the output in MaxCompute.

odps_column1	odps_column2
a	b
a	b
a	b

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
collectionName	The name of the replica set in MongoDB.	Yes	None
column	The columns in MongoDB. • name: the name of the column. • type: the data type of the column. • splitter: the delimiter. Specify this parameter only when you need to convert the string to an array. MongoDB supports arrays, but Data Integration does not. The array elements read by MongoDB are joined to a string by using this delimiter.	Yes	None
query	The filter condition for obtaining data from MongoDB. Only the time type is supported. For example, you can use the statement `"query":"{'operationTime': {'$gte':ISODate('${last_day}T00:00:00.424+0800')}}"` to obtain data where the time specified by operationTime is not earlier than 00:00 on the day specified by ${last_day}. In the preceding JSON string, ${last_day} is a scheduling parameter of DataWorks. The format is $[yyyy-mm-dd]. You can use comparison operators (such as $gt, $lt, $gte, and $lte), logical operators (such as $and and $or), and functions (such as max, min, sum, avg, and ISODate) supported by MongoDB as needed.	No	None

Configure MongoDB Reader by using the codeless UI

The codeless UI is not supported for MongoDB Reader.

Configure MongoDB Reader by using the code editor

In the following code, a node is configured to read data from a MongoDB database. For more information about the parameters, see the preceding parameter description.

```
{
   "type":"job",
   "version":"2.0",// The version number.
   "steps":[
        "reader": {
        "plugin": "mongodb", // The reader type.
        "parameter": {
          "datasource": "datasourceName", // The connection name.
```

```
"collectionName": "tag_data", // The name of the MongoDB collection.
"query":"",
"column": [
      {
         "name": "unique_id", // The field name.
         "type": "string" // The data type.
      },
      {
         "name": "sid",
         "type": "string"
      },
      {
         "name": "user_id",
         "type": "string"
      },
      {
         "name": "auction_id",
         "type": "string"
      },
      {
         "name": "content_type",
         "type": "string"
      },
      {
         "name": "pool_type",
         "type": "string"
      },
      {
         "name": "frontcat_id",
         "type": "array",
         "splitter": ""
      },
      {
         "name": "categoryid",
         "type": "array",
         "splitter": ""
      },
      {
         "name": "gmt_create",
         "type": "string"
      }
```

```
                },
                {
                  "name": "taglist",
                  "type": "array",
                  "splitter": " "
                },
                {
                  "name": "property",
                  "type": "string"
                },
                {
                  "name": "scorea",
                  "type": "int"
                },
                {
                  "name": "scoreb",
                  "type": "int"
                },
                {
                  "name": "scorec",
                  "type": "int"
                },
        {
                   "name": "a.b",
                   "type": "document.int"
                },
                {
                   "name": "a.b.c",
                   "type": "document.array",
                   "splitter": " "
                }
          ]
        }
      },
      {
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
```

```
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

Note You cannot retrieve data elements from arrays.

2.4.5.3.6. Configure Db2 Reader

This topic describes the data types and parameters supported by Db2 Reader and how to configure it by using the code editor.

Db2 Reader allows you to read data from Db2. Db2 Reader connects to a remote Db2 database and runs a SELECT statement to select and read data from the database.

Specifically, Db2 Reader connects to a remote Db2 database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The Db2 database runs the statement and returns the result. Then, Db2 Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- Db2 Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Db2 database.
- If you specify the querySql parameter, Db2 Reader directly sends the value of this parameter to the Db2 database.

Db2 Reader supports most Db2 data types. Make sure that your data types are supported.

The following table lists the data types supported by Db2 Reader.

Category	Db2 data type
Integer	SMALLINT
Floating point	DECIMAL, REAL, and DOUBLE
String	CHAR, CHARACTER, VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARCHAR, CLOB, LONG VARGRAPHIC, and DBCLOB
Date and time	DATE, TIME, and TIMESTAMP
Boolean	N/A
Binary	BLOB

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
jdbcUrl	The JDBC URL for connecting to the Db2 database. In accordance with official Db2 specifications, the URL must be in the `jdbc:db2://ip:port/database` format. You can also specify the information of the attachment facility.	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by Db2, for example, `["id", "1", "'const name'", "null", "upper('abc_lower')", "2.3", "true"]`. ○ id: a column name. ○ 1: an integer constant. ○ 'const name': a string constant, which is enclosed in single quotation marks (' '). ○ null: a null pointer. ○ upper('abc_lower'): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None
splitPk	The field used for data sharding when Db2 Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, Db2 Reader returns an error.	No	""
where	The WHERE clause. Db2 Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `gmt_create>$bizdate`. You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None

Parameter	Description	Required	Default value
querySql	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, Db2 Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. Note A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.	No	*1024*

Configure Db2 Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Db2 Reader.

Configure Db2 Reader by using the code editor

In the following code, a node is configured to read data from a Db2 database.

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"db2", // The reader type.
      "parameter":{
        "password":"", // The password for connecting to the database.
        "jdbcUrl":"", // The JDBC URL for connecting to the Db2 database.
        "column":[
          "id"
        ],
        "where":"", // The WHERE clause.
        "splitPk":"", // The field used for data sharding. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter.
        "table":"", // The name of the table to be synchronized.
        "username":"" // The username for connecting to the database.
```

```
        },
        "name":"Reader",
        "category":"reader"
      },
      {
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0" // The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary Db2 database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

Db2 is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. Db2 Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, Db2 Reader cannot obtain the new data.

Data consistency cannot be guaranteed when you enable Db2 Reader to run concurrent threads on a single sync node.

Db2 Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

- Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
- Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

  Db2 Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Db2 Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, Db2 Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  Db2 Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.4.5.3.7. Configure MySQL Reader

This topic describes the data types and parameters supported by MySQL Reader and how to configure it by using the codeless user interface (UI) and code editor.

MySQL Reader connects to a remote MySQL database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The MySQL database runs the statement and returns the result. Then, MySQL Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

In short, MySQL Reader connects to a remote MySQL database and runs a SELECT statement to select and read data from the database.

MySQL Reader can read tables and views. For table fields, you can specify all or some of the columns in sequence, adjust the column order, specify constant fields, and configure MySQL functions, such as now().

Data types

The following table lists the data types supported by MySQL Reader.

Category	MySQL data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, and BIGINT
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, TIME, and YEAR
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Note

- Data types that are not listed in the table are not supported.
- MySQL Reader considers tinyint(1) as the INTEGER type.
- Currently, MySQL Reader does not support MySQL 8.0 or later.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id","table","1","'mingya.wmy'","'null'","to_char(a+1)","2.3","true"]`. ◦ id: a column name. ◦ table: the name of a column that contains reserved keywords. ◦ 1: an integer constant. ◦ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ◦ null: ▪ " " indicates an empty value. ▪ null indicates a null value. ▪ 'null' indicates the string null. ◦ to_char(a + 1): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when MySQL Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, MySQL Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. For example, set this parameter to `gmt_create>$bizdate`. • You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized. • Do not set the where parameter to limit 10, which does not conform to the constraints of MySQL on the SQL WHERE clause.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. The priority of the querySql parameter is higher than those of the table, column, where, and splitPk parameters. If you specify the querySql parameter, MySQL Reader ignores the table, column, where, and splitPk parameters that you have configured. The datasource parameter parses information, including the username and password, from this parameter.	No	None

Parameter	Description	Required	Default value
singleOrMulti (applicable only to database and table sharding)	Specifies whether to shard the database or table. After you switch from the codeless UI to the code editor, the following configuration is automatically generated: `"singleOrMulti":"multi"`. However, if you use the code editor since the beginning, the configuration is not automatically generated and you must manually specify this parameter. If you do not specify this parameter or leave it empty, MySQL Reader can only read data from the first shard.	Yes	*multi*

Configure MySQL Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. **Note** The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.

Parameter	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MySQL Reader by using the code editor

In the following code, a node is configured to read data from a database or table that is not sharded. For more information about the parameters, see the preceding parameter description.

```
{
```

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"mysql", // The reader type.
      "parameter":{
        "column":[ // The columns to be synchronized.
          "id"
        ],
        "connection":[
          {  "querysql":["select a,b from join1 c join join2 d on c.id = d.id;"], // Specify the querySql parameter in the connection parameter as a string.
            "datasource":"", // The connection name.
            "table":[
              "xxx" // The name of the table to be synchronized.
            ]
          }
        ],
        "where":"", // The WHERE clause.
        "splitPk":"", // The shard key.
        "encoding":"UTF-8" // The encoding format.
      },
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0" // The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
```

```
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

In the following code, a node is configured to read data from a database or table that is sharded. For more information about the parameters, see the preceding parameter description.

> **Note** In the case of database and table sharding, MySQL Reader can read multiple MySQL tables with the same schema.

```
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "reader": {
      "plugin": "mysql",
      "parameter": {
        "connection": [
          {
            "table": [
              "tbl1",
              "tbl2",
              "tbl3"
            ],
            "datasource": "datasourceName1"
          },
          {
            "table": [
              "tbl4",
              "tbl5",
              "tbl6"
            ],
            "datasource": "datasourceName2"
          }
        ],
        "singleOrMulti": "multi",
        "splitPk": "db_id",
        "column": [
          "id", "name", "age"
        ],
        "where": "1 < id and id < 100"
      }
    },
    "writer": {
    }
  }
}
```

2.4.5.3.8. Configure Oracle Reader

This topic describes the data types and parameters supported by Oracle Reader and how to configure it by using the codeless user interface (UI) and code editor.

Oracle Reader allows you to read data from Oracle. Oracle Reader connects to a remote Oracle database and runs a SELECT statement to select and read data from the database.

Specifically, Oracle Reader connects to a remote Oracle database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The Oracle database runs the statement and returns the result. Then, Oracle Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- Oracle Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Oracle database.
- If you specify the querySql parameter, Oracle Reader directly sends the value of this parameter to the Oracle database.

Data types

Oracle Reader supports most Oracle data types. Make sure that your data types are supported.

The following table lists the data types supported by Oracle Reader.

Category	Oracle data type
Integer	NUMBER, ROWID, INTEGER, INT, and SMALLINT
Floating point	NUMERIC, DECIMAL, FLOAT, DOUBLE PRECISION, and REAL
String	LONG, CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, CHARACTER, CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING, and NCHAR VARYING
Date and time	TIMESTAMP and DATE
Boolean	BIT and BOOLEAN
Binary	BLOB, BFILE, RAW, and LONG RAW

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in JSON format. `["id", "1", "'mingya.wmy'", "null", "to_char(a + 1)", "2.3", "true"]` ◦ id: a column name. ◦ 1: an integer constant. ◦ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ◦ null: a null pointer. ◦ to_char(a + 1): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must be specified.	Yes	None
splitPk	The field used for data sharding when Oracle Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • The data types supported by the splitPk parameter include INTEGER, STRING, FLOAT, and DATE. • If you do not specify the splitPk parameter or leave it empty, Oracle Reader synchronizes data through a single thread.	No	None

Parameter	Description	Required	Default value
where	The WHERE clause. Oracle Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to row_number() or `id>2 and sex=1`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, Oracle Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. Note A value greater than 2048 may lead to out of memory (OOM) during the data synchronization process.	No	1024

Configure Oracle Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.

Parameter	Description
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. Note The Shard Key parameter is displayed only when you configure the source connection for a sync node.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Oracle Reader by using the code editor

In the following code, a node is configured to read data from an Oracle database.

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"oracle",
      "parameter":{
        "fetchSize":1024, // The number of data records to read at a time.
        "datasource":"", // The connection name.
        "column":[ // The columns to be synchronized.
          "id",
          "name"
        ],
        "where":"", // The WHERE clause.
        "splitPk":"", // The shard key.
        "table":"" // The name of the table to be synchronized.
      },
      "name":"Reader",
      "category":"reader"
    },
    {// The following template is used to configure Stream Writer. For more information about how to
```

```
configure other writers, see the corresponding topic.
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0" // The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
} "to":"Writer"
        }
      ]
    }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary Oracle database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

Oracle is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. Oracle Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, Oracle Reader cannot obtain the new data.

Data consistency cannot be guaranteed when you enable Oracle Reader to run concurrent threads on a single sync node.

Oracle Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

- Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
- Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

  Oracle Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  Oracle Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, Oracle Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  Oracle Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.4.5.3.9. Configure OSS Reader

This topic describes the data types and parameters supported by OSS Reader and how to configure it by using the codeless UI and code editor.

OSS Reader can read data stored in OSS. OSS Reader connects to OSS by using the official OSS Java SDK, reads data from OSS, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

OSS stores unstructured data only. OSS Reader supports the following features:

- Reads TXT objects that store logical two-dimensional tables. OSS Reader can read only TXT objects.
- Reads data stored in formats similar to CSV with custom delimiters.
- Reads data of various types as strings and supports constants and column pruning.
- Supports recursive reading and object name-based filtering.
- Supports the following object compression formats: GZIP, BZIP2, and ZIP.

  > **Note** You cannot compress multiple objects into one package.

- Reads multiple objects concurrently.

OSS Reader does not support the following features:

- Uses concurrent threads to read an uncompressed object.
- Uses concurrent threads to read a compressed object.

OSS Reader supports the following OSS data types: Bigint, Double, String, Datatime, and Boolean.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
Object	The name of the OSS object to read. You can specify multiple object names. For example, if a bucket has a directory named yunshi and this directory contains an object named ll.txt, you can set this parameter to yunshi/ll.txt. • If you specify a single OSS object, OSS Reader uses only one thread to read the object. Concurrent multi-thread reading of a single uncompressed object is coming soon. • If you specify multiple OSS objects, OSS Reader uses multiple threads to read these objects. The actual number of threads is determined by the number of channels. • When a name contains a wildcard, OSS Reader attempts to read all objects that match the name. For example, if you set the value to abc[0-9], OSS Reader reads objects abc0 to abc9. We recommend that you do not use wildcards because wildcards may cause out of memory (OOM). For more information, see OSS documentation. Note • Data Integration considers all the objects on a sync node as a single table. Make sure that all the objects on each sync node can adapt to the same schema. • Control the number of objects stored in a single directory. If a directory contains excessive objects, an OOM error may be returned. In this case, store the objects in different directories and then synchronize data.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, OSS Reader reads all data as strings. You can specify the column parameter in the following way: `json` `"column": ["*"]` You can also specify the column parameter in the following way: `json` `"column":` `{` `"type": "long",` `"index": 0 // The first INT-type column of the source object.` `},` `{` `"type": "string",` `"value": "alibaba" // The value of the current column. In this case, the value is a constant "alibaba."` `}` Note For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	By default, OSS Reader reads all data as strings.
fieldDelimiter	The column delimiter. Note You must specify the column delimiter for OSS Reader. The default delimiter is comma (,). The default setting for the column delimiter on the codeless UI is comma (,), too.	Yes	,

Parameter	Description	Required	Default value
compress	The compression format of the object. By default, this parameter is left empty, indicating that objects are not compressed. OSS Reader supports the following object compression formats: GZIP, BZIP2, and ZIP.	No	*By default, objects are not compressed.*
encoding	The encoding format of the object to read.	No	*utf-8*
nullFormat	The string that represents null. No standard strings can represent null in TXT objects. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat="null"`, Data Integration considers null as a null pointer. You can use the following formula to escape empty strings: \N=\\N.	No	None
skipHeader	Specifies whether to skip the header (if exists) of a CSV-like object. The skipHeader parameter is not supported for compressed objects.	No	*false*
csvReaderConfig	The configurations for reading CSV objects. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV objects, which supports many configurations.	No	None

Configure OSS Reader by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Object Name Prefix	The object parameter in the preceding parameter description. **Note** If an OSS object is named based on the date, for example, named as aaa/20171024abc.txt, you can set the object parameter to aaa/${bdp.system.bizdate}abc.txt.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
Encoding	The encoding parameter in the preceding parameter description. Default value: UTF-8.

Parameter	Description
Null String	The nullFormat parameter in the preceding parameter description. Enter a string that represents null. If the source connection contains the string, the string is replaced with null.
Compression Format	The compress parameter in the preceding parameter description. Default value: None.
Include Header	The skipHeader parameter in the preceding parameter description. Default value: No.

2. **Configure field mapping. It is equivalent to setting the column parameter provided in the preceding table.**

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click Add to add a field, or move the pointer over a field and click the Delete icon to delete the field.

Button	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. **Configure channel control policies.**

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure OSS Reader by using the code editor

In the following code, a node is configured to read data from OSS. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"oss",// The reader type.
      "parameter":{
        "nullFormat":"",// The string that represents null.
        "compress":"",// The compression format.
        "datasource":"",// The connection name.
        "column":[// The columns to be synchronized.
          {
            "index":0,// The ID of the column in the source table.
            "type":"string"// The data type.
          },
          {
            "index":1,
            "type":"long"
          },
          {
            "index":2,
            "type":"double"
          },
          {
            "index":3,
            "type":"boolean"
          },
          {
            "format":"yyyy-MM-dd HH:mm:ss", // The format of the time.
            "index":4,
            "type":"date"
          }
        ],
        "skipHeader":"", // Specifies whether to skip the header (if exists) of a CSV-like object.
        "encoding":"",// The encoding format.
        "fieldDelimiter":",",// The column delimiter.
        "fileFormat": "",// The format of the object saved by OSS Reader.
```

```
            "Object":[]// The name of the OSS object to read.
        },
        "name":"Reader",
        "category":"reader"
    },
    {
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
        "record":""// The maximum number of dirty data records allowed.
    },
    "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
    ]
  }
}
```

2.4.5.3.10. Configure FTP Reader

This topic describes the data types and parameters supported by File Transfer Protocol (FTP) Reader and how to configure it by using the codeless user interface (UI) and code editor.

FTP Reader allows you to read data from a remote FTP server. FTP Reader connects to an FTP server, reads data from the server, converts the data into a format that is readable by Data Integration, and then sends the converted data to a writer.

FTP Reader can read only FTP files that store logical two-dimensional tables, for example, text information in CSV format.

FTP servers store unstructured data only. Currently, FTP Reader supports the following features:

- Reads TXT files that store logical two-dimensional tables. FTP Reader can read only TXT files.
- Reads data stored in formats similar to CSV with custom delimiters.
- Reads data of various types as strings. Supports constants and column pruning.
- Supports recursive reading and file name-based filtering.
- Supports the following file compression formats: GZIP, BZIP2, ZIP, LZO, and LZO_DEFLATE.
- Reads multiple files concurrently.

Currently, FTP Reader does not support the following features:

- Uses concurrent threads to read an uncompressed file.
- Uses concurrent threads to read a compressed file.

The data types of remote FTP files are defined by FTP Reader.

Data Integration data type	FTP file data type
LONG	LONG
DOUBLE	DOUBLE
STRING	STRING
BOOLEAN	BOOLEAN
DATE	DATE

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
path	The path of the FTP file to read. You can specify multiple FTP file paths. • If you specify a single FTP file, FTP Reader uses only one thread to read the file. Concurrent multi-thread reading of a single uncompressed file is coming soon. • If you specify multiple FTP files, FTP Reader uses multiple threads to read these files. The actual number of threads is determined by the number of channels. • When a path contains a wildcard, FTP Reader attempts to read all files that match the path. If the path is ended with a slash (/), FTP Reader reads all files in the specified directory. For example, if you specify the path as /bazhen/, FTP Reader reads all files in the bazhen directory. Currently, FTP Reader only supports asterisks (*) as file name wildcards. Note • We recommend that you do not use asterisks (*) because this may cause out of memory (OOM) on a Java virtual machine (JVM). • Data Integration considers all the files on a sync node as a single table. Make sure that all the files on each sync node can adapt to the same schema and Data Integration has the permission to read all these files. • Make sure that the data format is similar to CSV. • An error occurs if no readable files exist in the specified path.	Yes	None

Parameter	Description	Required	Default value
column	The columns to read. The type parameter specifies the source data type. The index parameter specifies the ID of the column in the source table, starting from 0. The value parameter specifies the column value if the column is a constant column. By default, FTP Reader reads all data as strings. Specify this parameter as `"column":["*"]`. You can also specify the column parameter in the following way: `{ "type": "long", "index": 0 // The first INT-type column of the source file. }, { "type": "string", "value": "alibaba" // The value of the current column, that is, a constant "alibaba". }` For the column parameter, you must specify the type parameter and specify one of the index and value parameters.	Yes	By default, FTP Reader reads all data as strings.
fieldDelimiter	The column delimiter. Note You must specify the column delimiter for FTP Reader. The default delimiter is comma (,). The default setting for the column delimiter on the codeless UI is comma (,), too.	Yes	,
skipHeader	Specifies whether to skip the header (if exists) of a CSV-like file. The skipHeader parameter is not supported for compressed files.	No	false
encoding	The encoding format of the file to read.	No	*UTF-8*
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat:"null"`, Data Integration considers null as a null pointer.	No	None

Parameter	Description	Required	Default value
markDoneFileName	The name of the file used to indicate that the sync node can start. Data Integration checks whether the file exists before data synchronization. If the file does not exist, Data Integration checks again later. Data Integration starts the sync node only after the file is detected.	No	None
maxRetryTime	The maximum number of checks for the file used to indicate that the sync node can start. By default, 60 checks are allowed. Data Integration checks for the file every 1 minute. The whole process lasts at most 60 minutes.	No	*60*
csvReaderConfig	The configurations for reading CSV files. The parameter value must match the MAP type. A specific CSV reader is used to read data from CSV files, which supports many configurations.	No	None
fileFormat	The format of the file saved by FTP Reader. By default, FTP Reader converts the data into a two-dimensional table and stores the table in a CSV file. If you specify binary as the file format, Data Integration converts data into the binary format for replication and transmission. Generally, you need to specify this parameter only when you want to replicate the complete directory structure between storage systems such as FTP and Object Storage Service (OSS).	No	None

Configure FTP Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
File Path	The path parameter in the preceding parameter description.
File Type	The format of the file saved by FTP Reader. The default format is CSV.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
Encoding	The encoding parameter in the preceding parameter description. The default encoding format is *UTF-8*.
Null String	The nullFormat parameter in the preceding parameter description, which defines a string that represents the null value.

Parameter	Description
Compression Format	The compression format. By default, files are not compressed.
Include Header	The skipHeader parameter in the preceding parameter description. The default value is *No*.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Parameter	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. Configure channel control policies.

Parameter	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure FTP Reader by using the code editor

In the following code, a node is configured to read data from an FTP server.

```
{
```

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"ftp", // The reader type.
      "parameter":{
        "path":[], // The file path.
        "nullFormat":"", // The string that represents null.
        "compress":"", // The compression format.
        "datasource":"", // The connection name.
        "column":[ // The columns to be synchronized.
          {
            "index":0, // The ID of the column in the source table.
            "type":"" // The data type.
          }
        ],
        "skipHeader":"", // Specifies whether to skip the file header.
        "fieldDelimiter":",", // The column delimiter.
        "encoding":"UTF-8", // The encoding format.
        "fileFormat":"csv" // The format of the file saved by FTP Reader.
      },
      "name":"Reader",
      "category":"reader"
    },
    {// The following template is used to configure the writer. For more information, see the corresponding topic.
      "stepType":"stream",
      "parameter":{},
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0" // The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
```

```
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.3.11. Configure Table Store Reader

This topic describes the data types and parameters supported by Table Store Reader and how to configure it by using the code editor.

Table Store Reader can read incremental data from Table Store based on the specified range. Currently, Table Store Reader can read incremental data in the following ways:

- Reads data from the entire table.
- Reads data based on the specified range.
- Reads data from the specified shard.

Table Store is a NoSQL database service built on the Apsara distributed operating system that allows you to store and access large amounts of structured data in real time. Table Store organizes data into instances and tables. Using data sharding and load balancing technologies, Table Store seamlessly expands the data scale.

Table Store Reader connects to the Table Store server through the official Table Store Java SDK and reads data from the server. Then, Table Store Reader converts the data into a format that is readable by Data Integration based on the official data synchronization protocols, and sends the converted data to a writer.

Table Store Reader splits a sync node into concurrent tasks based on the table range to synchronize data in a Table Store table. Each thread is responsible for running a task.

Table Store Reader supports all Table Store data types. The following table lists the data types supported by Table Store Reader.

Category	Table Store data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN

Category	Table Store data type
Binary	BINARY

Note Table Store does not support data of the DATE type. Applications use the LONG-type UNIX timestamp to indicate the time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of the Table Store server.	Yes	None
accessId	The AccessKey ID for accessing Table Store.	Yes	None
accessKey	The AccessKey secret for accessing Table Store.	Yes	None
instanceName	The name of the Table Store instance. The instance is an entity for you to use and manage Table Store. After you activate the Table Store service, you must create an instance in the console before creating and managing tables. Instances are the basic unit for managing Table Store resources. All access control and resource measurement for applications are completed at the instance level.	Yes	None
table	The name of the source table. You can specify only one table as the source table. Multi-table synchronization is not required for Table Store.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be synchronized from the source table. The columns are described in a JSON array. Table Store is a NoSQL database service. You must specify column names for Table Store Reader to read data. • You can specify common columns. For example, you can specify {"name":"col1"} for Table Store Reader to read data in column 1. • You can specify certain columns to read. Table Store Reader only reads specified columns. • You can specify constant columns. For example, you can specify {"type":"STRING", "value":"DataX"} to read the column in which data is of the STRING type and the data value is DataX. The type parameter specifies the constant type. The supported types are STRING, INT, DOUBLE, BOOLEAN, BINARY, INF_MIN, and INF_MAX. If the constant type is BINARY, the constant value must be Base64-encoded. INF_MIN indicates the minimum value specified by Table Store, and INF_MAX indicates the maximum value specified by Table Store. If you set the type to INF_MIN or INF_MAX, do not set the value. Otherwise, errors may occur. • You cannot specify a function or custom expression, because Table Store does not provide functions or expressions similar to those of SQL. Table Store Reader cannot read columns that contain functions or expressions.	Yes	None
	The Table Store table range from which data is to be read. You can specify both or neither of the two parameters. The begin and end parameters define the value ranges of primary key columns in the Table Store table. Make sure that you specify the value ranges for all primary key columns in the table. If you do not need to limit a range, specify the parameters as {"type":"INF_MIN"} and {"type":"INF_MAX"}. For example, to read certain data from a Table Store table with the primary key of [DeviceID, SellerID], specify the begin and end parameters in the following way:		

Parameter	Description	Required	Default value
begin and end	"range": { "begin": [{"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INT", "value":"0"} // The minimum value of the SellerID field.], "end": [{"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INT", "value":"9999"} // The maximum value of the SellerID field.] } To read all data from the table, specify the begin and end parameters in the following way: "range": { "begin": [{"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INF_MIN"} // The minimum value of the SellerID field.], "end": [{"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INF_MAX"} // The maximum value of the SellerID field.] }		
split	The custom rule for data sharding. This parameter is an advanced setting. We recommend that you do not set this parameter. If data is unevenly distributed in a Table Store table and the automatic sharding feature of Table Store Reader fails to work, you can customize a sharding rule. The sharding rule specified by the split parameter must fall in the range specified by the begin and end parameters and must be the values of partition key columns. That is, you only need to specify the values of partition key columns instead of the values of primary key columns in the split parameter. To read data from a Table Store table with the primary key of [DeviceID, SellerID], specify the following parameters:		

Parameter	Description	Required	Default value
split	"range": { "begin": { {"type":"INF_MIN"}, // The minimum value of the DeviceID field. {"type":"INF_MIN"} // The minimum value of the SellerID field. }, "end": { {"type":"INF_MAX"}, // The maximum value of the DeviceID field. {"type":"INF_MAX"} // The maximum value of the SellerID field. }, // The specified sharding rule. If you specify a sharding rule, the sync node is split into concurrent tasks based on the values of the begin, end, and split parameters. Data is sharded only based on the partition key, that is, the first column of the primary key. // The data type of the partition key can be INF_MIN, INF_MAX, STRING, or INT. "split":[{"type":"STRING", "value":"1"}, {"type":"STRING", "value":"2"}, {"type":"STRING", "value":"3"}, {"type":"STRING", "value":"4"}, {"type":"STRING", "value":"5"}] }	No	None

Parameter	Description	Required	Default value

Configure Table Store Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Table Store Reader.

Configure Table Store Reader by using the code editor

In the following code, a node is configured to read data from a Table Store table.

```
{
  "type":"job",
  "version":"2.0", // The version number.
  "steps":[
    {
      "stepType":"ots", // The reader type.
      "parameter":{
        "datasource":"", // The connection name.
        "column":[ // The columns to be synchronized.
          {
            "name":"column1" // The name of the column.
          },
          {
            "name":"column2"
          },
          {
            "name":"column3"
          },
          {
            "name":"column4"
          },
          {
            "name":"column5"
          }
        ],
        "range":{
```

```
"split":[
  {
    "type":"INF_MIN"
  },
  {
    "type":"STRING",
    "value":"splitPoint1"
  },
  {
    "type":"STRING",
    "value":"splitPoint2"
  },
  {
    "type":"STRING",
    "value":"splitPoint3"
  },
  {
    "type":"INF_MAX"
  }
],
"end":[
  {
    "type":"INF_MAX"
  },
  {
    "type":"INF_MAX"
  },
  {
    "type":"STRING",
    "value":"end1"
  },
  {
    "type":"INT",
    "value":"100"
  }
],
"begin":[
  {
    "type":"INF_MIN"
  },
  {
```

```
                {
                    "type":"INF_MIN"
                },
                {
                    "type":"STRING",
                    "value":"begin1"
                },
                {
                    "type":"INT",
                    "value":"0"
                }
            ]
          },
          "table":"" // The name of the table to be synchronized.
        },
        "name":"Reader",
        "category":"reader"
      },
      {
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0" // The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1, // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
```

```
        }
      ]
    }
}
```

2.4.5.3.12. Configure PostgreSQL Reader

This topic describes the data types and parameters supported by PostgreSQL Reader and how to configure it by using the codeless UI and code editor.

PostgreSQL Reader connects to a remote PostgreSQL database and runs a SELECT statement to select and read data from the database. ApsaraDB for Relational Database Service (RDS) provides the PostgreSQL storage engine.

Specifically, PostgreSQL Reader connects to a remote PostgreSQL database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and sends the statement to the database. The PostgreSQL database runs the statement and returns the result. Then, PostgreSQL Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- PostgreSQL Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SQL statement to the PostgreSQL database.
- If you specify the querySql parameter, PostgreSQL Reader directly sends the value of this parameter to the PostgreSQL database.

Data types

PostgreSQL Reader supports most PostgreSQL data types. Ensure that your data types are supported.

The following table lists the data types supported by PostgreSQL Reader.

Category	PostgreSQL data type
Integer	bigint, bigserial, integer, smallint, and serial
Float	double, precision, money, numeric, and real
String	varchar, char, text, bit, and inet
Date and time	date, time, and timestamp
Boolean	boolean
Binary	bytea

> **Note**
>
> - Except for the preceding data types, other types are not supported.
> - You need to convert the money, inet, and bit types by using syntax such as `a_inet::varchar`.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None
column	An array of columns to be synchronized from the configured table, in JSON format. The default value is [*], which indicates all columns. • Column pruning is supported, which means that you can select and export specific columns. • Change of the column order is supported, which means that you can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with SQL syntax supported by MySQL. For example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in a pair of single quotation marks ('). ○ 'null': a string. ○ to_char(a + 1): a function expression. ○ 2.3: a float value. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. It cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when PostgreSQL Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then initiates concurrent data synchronization threads, which improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, float, and date. If you specify this parameter to a column of an unsupported type, PostgreSQL Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. PostgreSQL Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to `id>2 and sex=1`. • The WHERE clause can be used for synchronizing incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you need to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, PostgreSQL Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. Note A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process.	No	*512*

Configure PostgreSQL Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the data synchronization node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected data store.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column. Only integer fields are supported. If data sharding is performed based on the configured shard key, data can be read concurrently to improve data synchronization efficiency. Note The Shard Key parameter is displayed only when you configure the source connection for a data synchronization node.

2. Configure field mapping (the column parameter in the preceding parameter description).

   Fields in the source table (left) have a one-to-one mapping with fields in the destination table (right). You can click **Add** to add a field or move the pointer over a field and click the **Delete** icon to delete a field.

Configuration item	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	The fields are automatically sorted based on specified rules.
Change Fields	You can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, while other blank rows are ignored.

Configuration item	Description
Add	○ You can enter constants. Each constant must be enclosed in single quotation marks ('), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ If the value you entered cannot be parsed, the type is displayed as Unidentified.

3. Configure the channel.

Parameter	Description
DMU	The billing unit of Data Integration. Note Use caution while setting the DMU parameter. Its value limits the maximum number of concurrent threads.
Concurrent Threads	The maximum number of concurrent data synchronization threads. If you specify this parameter, the data records are split based on the shard key specified for the reader so that they are synchronized in multiple threads concurrently. This improves the transmission rate.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the data synchronization node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the data synchronization node on the custom resource group. Set the resource group properly based on network conditions of the data stores, resource group usage, and business importance.

Configure PostgreSQL Reader by using the code editor

In the following code, a node is configured to read data from a PostgreSQL database.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"postgresql",// The reader type
```

```
        "stepType":"postgresql",// The reader type.
        "parameter":{
          "datasource":"",// The connection name.
          "column":[// The columns to be synchronized.
            "col1",
            "col2"
          ],
          "where":"",// The WHERE clause.
          "splitPk":"",// The shard key based on which the table is sharded. Data Integration initiates c
oncurrent threads to synchronize data.
          "table":""// The name of the table to be synchronized.
        },
        "name":"Reader",
        "category":"reader"
      },
      { // The following template is used to configure the writer. For more information, see the documen
t of the corresponding writer.
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false,// A value of false indicates that the bandwidth is not throttled. A value of true
indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set t
his parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
        "dmu":1// The DMU value.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
```

```
    ]
  }
}
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary PostgreSQL database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  PostgreSQL is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a data synchronization node starts. PostgreSQL Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, the reader cannot obtain the new data.

  Data consistency cannot be ensured when you enable PostgreSQL Reader to run concurrent threads on a single data synchronization node.

  PostgreSQL Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions, and they read data at different times. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single data synchronization node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single data synchronization node. Essentially, do not specify the splitPk parameter. In this way, data consistency is ensured while data is synchronized at a low efficiency.
  - Disable writers to ensure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services can be interrupted.

- Character encoding

  A PostgreSQL database supports only EUC_CN and UTF-8 encoding formats for simplified Chinese characters. PostgreSQL Reader uses JDBC, which can automatically convert encoding of characters. Therefore, you do not need to specify the encoding.

  If data is written to the PostgreSQL database in an encoding format different from that specified by the PostgreSQL database, PostgreSQL Reader cannot recognize this inconsistency and may export garbled characters.

- Incremental data synchronization

  PostgreSQL Reader connects to a database through JDBC and uses `a SELECT statement with a WHERE clause` to read incremental data in either of the following ways:

  - For batch data, incremental add, update, and delete operations (including logical delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, PostgreSQL Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  PostgreSQL Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.4.5.3.13. Configure SQL Server Reader

This topic describes the data types and parameters supported by SQL Server Reader and how to configure it by using the codeless user interface (UI) and code editor.

SQL Server Reader connects to a remote SQL Server database and runs a SELECT statement to select and read data from the database.

Specifically, SQL Server Reader connects to a remote SQL Server database through Java Database Connectivity (JDBC), generates a SELECT statement based on your configurations, and then sends the statement to the database. The SQL Server database runs the statement and returns the result. Then, SQL Server Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- SQL Server Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the SQL Server database.
- If you specify the querySql parameter, SQL Server Reader directly sends the value of this parameter to the SQL Server database.

SQL Server Reader supports most SQL Server data types. Make sure that your data types are supported.

The following table lists the data types supported by SQL Server Reader.

Category	SQL Server data type
Integer	bigint, int, smallint, and tinyint
Floating point	float, decimal, real, and numeric
String	char, nchar, ntext, nvarchar, text, varchar, nvarchar (max), and varchar (max)
Date and time	date, datetime, and time
Boolean	bit
Binary	binary, varbinary, varbinary (max), and timestamp

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id", "table","1", "'mingya.wmy'", "'null'", "to_char(a+1)", "2.3", "true"]`. ◦ id: a column name. ◦ table: the name of a column that contains reserved keywords. ◦ 1: an integer constant. ◦ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ◦ 'null': a string. ◦ to_char(a + 1): a function expression. ◦ 2.3: a floating-point constant. ◦ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when SQL Server Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, SQL Server Reader returns an error.	No	None
where	The WHERE clause. SQL Server Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to limit 10 during a test. For example, if you need to synchronize data generated on the current day, set this parameter to `gmt_create > $bizdate`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is synchronized.	No	None
querySql	The SELECT statement used for refined data filtering. Specify this parameter in the following format: `"querysql" : "SELECT statement",`. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. If you specify the querySql parameter, SQL Server Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. Note A value larger than 2048 may lead to the out of memory (OOM) error during the data synchronization process.	No	*1024*

Configure SQL Server Reader by using the codeless UI

1. Configure the connections.

   Configure the source and destination connections for the sync node.

Configuration item	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and enter the name of a connection that has been configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Filter	The filter condition for the data to be synchronized. Currently, filtering based on the limit keyword is not supported. The SQL syntax is determined by the selected connection.
Shard Key	The shard key. You can use a column in the source table as the shard key. We recommend that you use the primary key or an indexed column.

2. Configure field mapping, that is, the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

Configuration item	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout. The fields are automatically sorted based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box that appears, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Configuration item	Description
Expected Concurrency	The maximum number of concurrent threads to read data from or write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure SQL Server Reader by using the code editor

In the following code, a node is configured to read data from an SQL Server database.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"sqlserver",// The reader type.
      "parameter":{
        "datasource":"",// The connection name.
        "column":[// The columns to be synchronized.
          "id",
          "name"
        ],
        "where":"",// The WHERE clause.
        "splitPk":"",// The shard key based on which the table is sharded.
        "table":""// The name of the table to be synchronized.
      },
      "name":"Reader",
      "category":"reader"
    },
    {// The following template is used to configure the writer. For more information, see the correspo
```

```
nding topic.
            "stepType":"stream",
            "parameter":{},
            "name":"Writer",
            "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

If you want to use the querySql parameter to specify a SELECT statement to query data, see the following sample code in the script of SQL Server Reader. Assume that the SQL Server connection is sql_server_source, the table to be queried is dbo.test_table, and the column to be queried is name.

```
{
  "stepType": "sqlserver",
  "parameter": {
    "querySql": "select name from dbo.test_table",
    "datasource": "sql_server_source",
    "column": [
      "name"
    ],
    "where": "",
    "splitPk": "id"
  },
  "name": "Reader",
  "category": "reader"
},
```

Additional instructions

- Data synchronization between primary and secondary databases

  A secondary SQL Server database can be deployed for disaster recovery. The secondary database continuously synchronizes data from the primary database based on binlogs. Especially when network conditions are unfavorable, data latency between the primary and secondary databases is unavoidable, which can lead to data inconsistency.

- Concurrency control

  SQL Server is a relational database management system (RDBMS), which supports strong consistency for data queries. A database snapshot is created before a sync node starts. SQL Server Reader reads data from the database snapshot. Therefore, if new data is written to the database during data synchronization, the reader cannot obtain the new data.

  Data consistency cannot be guaranteed when you enable SQL Server Reader to run concurrent threads on a single sync node.

  SQL Server Reader shards the table based on the splitPk parameter and runs multiple concurrent threads to synchronize data. These concurrent threads belong to different transactions. They read data at different time points. This means that the concurrent threads observe different snapshots.

  Theoretically, the data inconsistency issue is unavoidable if a single sync node includes multiple threads. However, two workarounds are available:

  - Do not enable concurrent threads on a single sync node. Essentially, do not specify the splitPk parameter. In this way, data consistency is guaranteed although data is synchronized at a low efficiency.
  - Disable writers to make sure that the data is unchanged during data synchronization. For example, lock the table and disable data synchronization between primary and secondary databases. In this way, data is synchronized efficiently but your ongoing services may be interrupted.

- Character encoding

SQL Server Reader uses JDBC, which can automatically convert the encoding of characters. Therefore, you do not need to specify the encoding format.

- Incremental data synchronization

  SQL Server Reader connects to a database through JDBC and uses a SELECT statement with a WHERE clause to read incremental data in the following ways:

  - For data in batches, incremental add, update, and delete operations (including logically delete operations) are distinguished by timestamps. Specify the WHERE clause based on the timestamp. The timestamp must be later than the latest timestamp in the last synchronization.
  - For streaming data, specify the WHERE clause based on the data record ID. The data record ID must be larger than the maximum ID involved in the last synchronization.

  If incremental data cannot be distinguished, SQL Server Reader cannot perform incremental synchronization but can perform full synchronization only.

- Syntax validation

  SQL Server Reader allows you to specify custom SELECT statements by using the querySql parameter but does not verify the syntax of the custom SELECT statements.

2.4.5.3.14. Configure LogHub Reader

This topic describes the data types and parameters supported by LogHub Reader and how to configure it by using the codeless UI and code editor.

As an all-in-one real-time data logging service, Log Service provides features to collect, consume, deliver, query, and analyze log data. It can comprehensively improve the capabilities to process and analyze numerous logs. LogHub Reader consumes real-time log data in LogHub by using the Java SDK for Log Service, converts the data to a format that can be read by the Data Integration service, and sends the converted data to a writer.

How it works

LogHub Reader consumes real-time log data in LogHub by using the following version of the Java SDK for Log Service:

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>aliyun-log</artifactId>
    <version>0.6.7</version>
</dependency>
```

In Log Service, Logstore is a basic unit for collecting, storing, and querying log data. The read and write logs of a Logstore are stored in a shard. Each Logstore consists of several shards, each of which is defined by a left-closed and right-open interval of MD5 values so that intervals do not overlap each other. The range of all intervals covers all the allowed MD5 values. Each shard can independently provide some services.

- Write: 5 Mbit/s, 2,000 times/s.
- Read: 10 Mbit/s, 100 times/s.

LogHub Reader consumes log data in shards by following this process in which the GetCursor and BatchGetLog API operations are called:

- Obtain a cursor based on the time range.
- Read logs based on the cursor and step parameters and return the next cursor.
- Keep moving the cursor to consume logs.
- Split the node to concurrent threads based on shards.

Data types

The following table lists the data types supported by LogHub Reader.

Data Integration data type	LogHub data type
STRING	STRING

Parameters

Parameter	Description	Required	Default value
endpoint	The Log Service endpoint, which is a URL for accessing a project and log data. It varies based on the Alibaba Cloud region where the project resides and the project name.	Yes	None
accessId	The AccessKey ID for connecting to Log Service.	Yes	None
accessKey	The AccessKey secret for connecting to Log Service.	Yes	None
project	The name of the project. A project is the basic unit for managing resources in Log Service. You can exercise access control at the project level, and isolate resources among different projects.	Yes	None
logstore	The name of the Logstore. A Logstore is the basic unit for collecting, storing, and querying log data in Log Service.	Yes	None
batchSize	The number of entries queried from Log Service at a time.	No	*128*
column	The column name in each log entry. You can configure a column that stores metadata in a source table of LogHub in such a way that the metadata in this column is inserted into the destination table. Supported metadata includes the log topic, unique identifier of the collection machine, host name, path, and log time. Note The column name is case-sensitive.	Yes	None

Parameter	Description	Required	Default value
beginDateTime	The start time of data consumption. The value is the time when log data arrives at LogHub. This parameter defines the left boundary of a left-closed and right-open interval in the format of yyyyMMddHHmmss, for example, 20180111013000. The parameter can work with the scheduling time parameter in DataWorks. Note The beginDateTime and endDateTime parameters must be used in pairs.	You must specify either beginDateTime or beginTimestampMillis, but not both.	None
endDateTime	The end time of data consumption in the format of yyyyMMddHHmmss, such as 20180111013010. This parameter defines the right boundary of a left-closed and right-open interval and can work with the scheduling time parameter in DataWorks. Note Make sure that the time specified by the endDateTime parameter of the previous interval is the same as or later than the time specified by the beginDateTime parameter of the current interval. If the intervals do not overlap, data may fail to be read in some regions.	You must specify either endDateTime or endTimestampMillis, but not both.	None
beginTimestampMillis	The start time of data consumption. This parameter specifies the left boundary of the left-closed and right-open interval, measured in milliseconds. Note The beginTimestampMillis and endTimestampMillis parameters must be used in pairs. A value of -1 indicates the position where the cursor starts in Log Service, which is specified by CursorMode.BEGIN. We recommend that you specify the beginDateTime parameter.	You must specify either beginTimestampMillis or beginDateTime, but not both.	None

Parameter	Description	Required	Default value
endTimestampMillis	The end time of data consumption, measured in milliseconds. This parameter defines the right boundary of the left-closed and right-open interval. Note The endTimestampMillis and beginTimestampMillis parameters must be used in pairs. A value of -1 indicates the position where the cursor ends in Log Service, which is specified by CursorMode.END. We recommend that you specify the endDateTime parameter.	You must specify either endTimestampMillis or endDateTime, but not both.	None

Configure LogHub Reader in the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The type and name of the source connection. Select a connection that you have configured in DataWorks.
Logstore	The name of the Logstore from which data is read.
Start Timestamp	The start time of data consumption. The value is the time when log data arrives at LogHub. This parameter defines the left boundary of a left-closed and right-open interval in the format of yyyyMMddHHmmss, for example, 20180111013000. The parameter can work with the scheduling time parameter in DataWorks.
End Timestamp	The end time of data consumption in the format of yyyyMMddHHmmss, such as 20180111013010. This parameter defines the right boundary of a left-closed and right-open interval and can work with the scheduling time parameter in DataWorks.
Records per Batch	The number of entries queried from Log Service at a time.

2. Configure field mapping. It is equivalent to setting the column parameter provided in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), such as 'abc' and '123'. ○ You can use scheduling parameters, such as ${bizdate}. ○ You can enter functions supported by relational databases, such as now() and count(1). ○ Fields that cannot be parsed are indicated by Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure LogHub Reader by using the code editor

In the following code, a node is configured to read data from LogHub. For more information about the parameters, see the preceding parameter description.

```
{
 "type":"job",
 "version":"2.0",// The version number.
 "steps":[
   {
     "stepType":"loghub",// The reader type.
     "parameter":{
       "datasource":"",// The connection name.
       "column":[// The columns to be synchronized.
         "col0",
         "col1",
         "col2",
         "col3",
         "col4",
         "=Topic",// The log topic.
         "HostName",// The hostname.
         "Path",// The path.
         "LogTime"// The log time.
       ],
       "beginDateTime":"",// The start time of data consumption.
       "batchSize":"",// The number of entries that are queried from Log Service at a time.
       "endDateTime":"",// The end time of data consumption.
       "fieldDelimiter":",",// The column delimiter.
       "encoding":"UTF-8",// The encoding format.
       "logstore":"// The name of the target Logstore.
     },
     "name":"Reader",
     "category":"reader"
   },
   {
```

```
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false, // Specifies whether to enable bandwidth throttling. A value of false indicates th
at the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxim
um transmission rate takes effect only if you set this parameter to true.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

Note If the metadata in JSON format is prefixed by tag, delete the tag prefix. For example, change `__tag__:__client_ip__` to `__client_ip__`.

2.4.5.3.15. Configure Tablestore Reader-Internal

This topic describes the data types and parameters supported by Tablestore Reader-Internal and how to configure it by using the code editor.

Tablestore is a NoSQL database service built on the Apsara distributed operating system that allows you to store and access large amounts of structured data in real time. Tablestore organizes data into instances and tables. It can seamlessly expand the data scale by using data sharding and load balancing technologies.

Tablestore Reader-Internal is used to export data for the Tablestore Internal model, whereas Tablestore Reader is used to export data for the Tablestore Public model.

Tablestore Reader-Internal can export data in multi-version mode or normal mode:

- Multi-version mode: Tablestore stores multiple versions of column values, and this mode allows you to export data of multiple versions.

  Tablestore Reader-Internal converts a cell to a 4-tuple of a one-dimensional table: PrimaryKey (columns 1 to 4), ColumnName, Timestamp, and Value. This process is similar to that for the multi-version mode of HBase Reader. Each {PrimaryKey, ColumnName, Timestamp, Value} tuple is sent to a writer as four columns in Data Integration records.
- Normal mode: This mode allows you to export the latest version of each column in each row, which is the same as the normal mode of HBase Reader. For more information, see the normal mode of HBase Reader in Configure an HBase connection.

Tablestore Reader-Internal connects to a Tablestore server by using the official Java SDK for Tablestore and reads data from the server. Tablestore Reader-Internal optimizes the read process by providing features such as performing retry attempts when a timeout or exception occurs.

Tablestore Reader-Internal supports all Tablestore data types. The following table lists the data types supported by Tablestore Reader-Internal.

Data Integration data type	Tablestore data type
LONG	INTEGER
DOUBLE	DOUBLE
STRING	STRING
BOOLEAN	BOOLEAN
BYTES	BINARY

Parameters

Parameter	Description	Required	Default value
mode	The mode in which Tablestore Reader-Internal exports data. Valid values: *normal* and *multiVersion*.	Yes	None
endpoint	The endpoint of the Tablestore server.	Yes	None
accessId	The AccessKey ID for connecting to Tablestore.	Yes	None
accessKey	The AccessKey secret for connecting to Tablestore.	Yes	None

Parameter	Description	Required	Default value
instanceName	The name of the Tablestore instance. The instance is an entity for you to use and manage Tablestore. After you activate the Tablestore service, you must create an instance in the console before you create and manage tables. Instances are the basic units for managing Tablestore resources. All access control and resource measurement for applications are implemented at the instance level.	Yes	None
table	The name of the source table. You can specify only one table as the source table. Multi-table synchronization is not required for Tablestore.	Yes	None
range	The range of the data to export, in the format of [begin,end). • If the value of the begin parameter is smaller than that of the end parameter, data is read in forward order. • If the value of the begin parameter is larger than that of the end parameter, data is read in reverse order. • The value of the begin parameter cannot be the same as that of the end parameter. • The following value types are supported: STRING, INT, and BINARY. Binary data is passed in as Base64 strings in binary format. INF_MIN represents an infinitely small value and INF_MAX represents an infinitely large value.	No	By default, data is read from the beginning of the table to the end of the table.

Parameter	Description	Required	Default value
range: {"begin"}	The start of the data to export. Enter an empty array, a primary key prefix, or a complete primary key. In forward order, the default primary key suffix is INF_MIN. In reverse order, the default primary key suffix is INF_MAX. This parameter specifies the value range of the Tablestore primary key and is used for data filtering. If you do not specify this parameter, the minimum value is used by default. The JSON format does not support binary data. If the data type of the PrimaryKey column is BINARY, you must use the Java method Base64.encodeBase64String to convert binary data to a string, and then enter the string as the value of the parameter. Example: • `byte[] bytes = "hello".getBytes();` : constructs binary data, which is the byte value of the string hello. • `String inputValue = Base64.encodeBase64String(bytes)` : calls the Base64.encodeBase64String method to convert the binary data to a string. After you run the preceding code, the string `"aGVsbG8="` is returned for the inputValue parameter. Finally, set this parameter to `{"type":"binary","value" : "aGVsbG8="}` .	No	Data is read from the beginning of the table.
range: {"end "}	The end of the data to export. Enter an empty array, a primary key prefix, or a complete primary key. In forward order, the default primary key suffix is INF_MAX. In reverse order, the default primary key suffix is INF_MIN. The JSON format does not support binary data. If the data type of the PrimaryKey column is BINARY, you must use the Java method Base64.encodeBase64String to convert binary data to a string, and then enter the string as the value of the parameter. Example: • `byte[] bytes = "hello".getBytes();` : constructs binary data, which is the byte value of the string hello. • `String inputValue = Base64.encodeBase64String(bytes)` : calls the Base64.encodeBase64String method to convert the binary data to a string. After you run the preceding code, the string `"aGVsbG8="` is returned for the inputValue parameter. Finally, set this parameter to `{"type":"binary", "value":"aGVsbG8="}` .	No	Data is read until the end of the table.

Parameter	Description	Required	Default value
range: {"split"}	If an excessively large amount of data needs to be exported, you can specify this parameter to split one node to multiple concurrent threads. Note • The field based on which the node is split must be the shard key, which is the first column of the primary key, and the data type of the field must be the same as that of the partition key. • The specified field must be within the value range that is specified by the begin and end parameters. • The values of this field must be sorted in the descending or ascending order based on the data reading order that is determined by values of the begin and end parameters.	No	No sharding rule is specified.
column	The columns to be exported. Both regular and constant columns can be exported. A regular column is in the format of `{"name":"{your column name}"}` . Note • Constant columns cannot be exported in multi-version mode. • You cannot specify the PrimaryKey column. The exported tuple data contains the complete primary key by default. • Each column can be exported only once.	None	All versions of all columns are exported.
timeRange (applicable only to the multi-version mode)	The time range of the requested data, in the format of [begin,end). Note The value of the begin parameter must be smaller than that of the end parameter.	No	All the data is read.
timeRange: {"begin"} (applicable only to the multi-version mode)	The start time for reading data. Valid values: 0 to LONG_MAX.	No	0

Parameter	Description	Required	Default value
timeRange: {"end"} (applicable only to the multi-version mode)	The end time for reading data. Valid values: 0 to LONG_MAX.	No	*LONG_MAX (922337203685477 5806L)*
maxVersion (applicable only to the multi-version mode)	The specified version of the requested data. Valid values: 1 to INT32_MAX.	No	The data of all versions is read.

Configure Tablestore Reader-Internal by using the codeless UI

The codeless UI is not supported for Tablestore Reader-Internal.

Configure Tablestore Reader-Internal by using the code editor

- Multi-version mode

```
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "reader": {
      "plugin": "otsreader-internalreader",
      "parameter": {
        "mode": "multiVersion",
        "endpoint": "",
        "accessId": "",
        "accessKey": "",
        "instanceName": "",
        "table": "",
        "range": {
          "begin": [
            {
              "type": "string",
              "value": "a"
            },
            {
              "type": "INF_MIN"
            }
```

```
            ],
            "end": [
              {
                "type": "string",
                "value": "g"
              },
              {
                "type": "INF_MAX"
              }
            ],
            "split": [
              {
                "type": "string",
                "value": "b"
              },
              {
                "type": "string",
                "value": "c"
              }
            ]
          },
          "column": [
            {
              "name": "attr1"
            }
          ],
          "timeRange": {
            "begin": 1400000000,
            "end": 1600000000
          },
          "maxVersion": 10
        }
      }
    },
    "writer": {}
}
```

- Normal mode

```
{
    "type": "job",
```

```
"version": "1.0",
"configuration": {
  "reader": {
    "plugin": "otsreader-internalreader",
    "parameter": {
      "mode": "normal",
      "endpoint": "",
      "accessId": "",
      "accessKey": "",
      "instanceName": "",
      "table": "",
      "range": {
        "begin": [
          {
            "type": "string",
            "value": "a"
          },
          {
            "type": "INF_MIN"
          }
        ],
        "end": [
          {
            "type": "string",
            "value": "g"
          },
          {
            "type": "INF_MAX"
          }
        ],
        "split": [
          {
            "type": "string",
            "value": "b"
          },
          {
            "type": "string",
            "value": "c"
          }
        ]
      },
```

```
            "column": [
              {
                "name": "pk1"
              },
              {
                "name": "pk2"
              },
              {
                "name": "attr1"
              },
              {
                "type": "string",
                "value": ""
              },
              {
                "type": "int",
                "value": ""
              },
              {
                "type": "double",
                "value": ""
              },
              {
                "type": "binary",
                "value": "aGVsbG8="
              }
            ]
          }
        }
    },
    "writer": {}
}
```

2.4.5.3.16. Configure OTSStream Reader

This topic describes the data types and parameters supported by OTSStream Reader and how to configure it by using the code editor.

OTSStream Reader is mainly used for exporting the incremental data of Table Store. Incremental data can be considered as operation logs that include data and operation information.

Unlike plug-ins for exporting full data, OTSStream Reader only supports the multi-version mode. You cannot export the data of specified columns when using OTSStream Reader for exporting incremental data. This restriction is related to the implementation of exporting incremental data. The following section describes the implementation process.

Before using OTSStream Reader, make sure that the Stream feature is enabled. You can enable this feature when creating the table or using the UpdateTable operation in the SDK.

The method for enabling Stream is described as follows:

```
SyncClient client = new SyncClient("", "", "", "");
Enable this feature when creating a table.
CreateTableRequest createTableRequest = new CreateTableRequest(tableMeta);
createTableRequest.setStreamSpecification(new StreamSpecification(true, 24)); // The value 24 indic
ates that the incremental data is retained for 24 hours.
client.createTable(createTableRequest);
If this feature is not enabled when the table is created, enable it by using the UpdateTable operation.
UpdateTableRequest updateTableRequest = new UpdateTableRequest("tableName");
updateTableRequest.setStreamSpecification(new StreamSpecification(true, 24));
client.updateTable(updateTableRequest);
```

Implementation

You can enable the Stream feature and set the expiration time by using the UpdateTable operation in the SDK. After the Stream feature is enabled, the Table Store server saves your operation logs additionally. Each partition has a sequential operation log queue. Each operation log is removed by garbage collection after the specified expiration time.

The Table Store SDK provides several Stream APIs for reading these operation logs. OTSStream Reader obtains incremental data by using these APIs, transforms incremental data into multiple 6-tuples (pk, colName, version, colValue, opType, and sequenceInfo), and imports them into MaxCompute.

Exported data format

In the multi-version mode of Table Store, table data is organized in a three-level architecture: row, column, and version. One row can have multiple columns. The column name is not fixed, and each column can have multiple versions. Each version has a specific timestamp (the version number).

You can perform read/write operations by using Table Store APIs. Table Store stores the incremental data by storing the records of recent write and modify operations on table data. Incremental data can be considered as a set of operation records.

Table Store supports the following three types of modify operations:

- PutRow: writes a row. If the row already exists, it is overwritten.
- UpdateRow: updates a row without changing other data of the original row. You can add column values, overwrite column values if the corresponding version of the column already exists, delete all the versions of a column, or delete a version of a column.
- DeleteRow: deletes a row.

Table Store generates incremental data records based on each type of operations. OTSStream Reader reads these records and exports the data in the format of DataX.

Table Store supports dynamic columns and the multi-version mode. Therefore, a row exported by OTSStream Reader corresponds to a version of a column rather than a row in Table Store. A row in Table Store may correspond to multiple exported rows. Each exported row includes the primary key value, column name, timestamp of the version for the column (version number), value of the version, and operation type. If the isExportSequenceInfo parameter is set to true, time series information is also included.

When the data is transformed into the DataX format, the following four types of operations are defined:

- U (UPDATE): Writes a version of a column.
- DO (DELETE_ONE_VERSION): Deletes a version of a column.
- DA (DELETE_ALL_VERSION): Deletes all the versions of a column. Delete all the versions of the corresponding column according to the primary key and the column name.
- DR (DELETE_ROW): Deletes a row. Delete all the data of the row according to the primary key.

In the following example, the table has two primary key columns: pkName1 and pkName2.

pkName1	pkName2	columnName	timestamp	columnValue	opType
pk1_V1	pk2_V1	col_a	1441803688001	col_val1	U
pk1_V1	pk2_V1	col_a	1441803688002	col_val2	U
pk1_V1	pk2_V1	col_b	1441803688003	col_val3	U
pk1_V2	pk2_V2	col_a	1441803688000	-	DO
pk1_V2	pk2_V2	col_b	-	-	DA
pk1_V3	pk2_V3	-	-	-	DR
pk1_V3	pk2_V3	col_a	1441803688005	col_val1	U

In this example, seven rows are exported, corresponding to three rows in the Table Store table. The primary keys for the three rows are (pk1_V1, pk2_V1), (pk1_V2, pk2_V2), and (pk1_V3, pk2_V3).

- For the row whose primary key is (pk1_V1, pk2_V1), three operations are included: writing two versions of column col_a and one version of column col_b.
- For the row whose primary key is (pk1_V2, pk2_V2), two operations are included: deleting one version of column col_a and deleting all versions of column col_b.
- For the row whose primary key is (pk1_V3, pk2_V3), two operations are included: deleting the row and writing one version of column col_a.

Data types supported by OTSStream Reader

Currently, OTSStream Reader supports all Table Store data types. The following table lists the data types supported by OTSStream Reader.

Category	OTSStream data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Parameters

Parameter	Description	Required	Default value
dataSource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
dataTable	The name of the table from which incremental data is exported. You must enable the Stream feature for a table when creating the table, or by calling the UpdateTable operation after creating the table.	Yes	None

Parameter	Description	Required	Default value
statusTable	The name of the table used by OTSStream Reader to store status records. These records help to filter out the data that is not covered by the target range and improve export efficiency. A statusTable is the table to store status records. If no such table exists, the reader automatically creates one. When a task of exporting batch data is completed, you do not need to delete the table. The status records in the table can be used for the next export task. • You do not need to manually create a statusTable. You only need to provide a table name. The reader attempts to create a statusTable under your instance. If no such table exists, the reader automatically creates one. If such a table already exists, the reader determines whether the Meta of the table is proper. If not, an exception is thrown. • When an export task is completed, you do not need to delete the table. The status of the table can be used for the next export task. • The table enables TTL and data expires automatically. Therefore, the data volume is small. • You can use a statusTable to store status records of multiple dataTables that are managed by the same instance. The status records are independent of each other. In conclusion, you must configure a name such as TableStoreStreamReaderStatusTable. Note that the name must not be the same as that for any business-related table.	Yes	None
startTimestampMillis	The start time (included) in milliseconds of the incremental data. • The Reader plugin finds a point corresponding to startTimestampMillis from the statusTable, and starts to read and export data from this point. • If the reader cannot find the corresponding point, it starts to read incremental data retained by the system from the first entry, and skip the data which is written later than startTimestampMillis.	No	None
endTimestampMillis	The end time (excluded) in milliseconds of the incremental data. • The reader exports data from the time specified by the startTimestampMillis parameter and ends at the data entry with a timestamp that is later than or equal to endTimestampMillis. • If the reader has read all the incremental data, it stops reading data even before the time specified by the endTimestampMillis parameter.	No	None

Parameter	Description	Required	Default value
date	The date when data is exported. The format is yyyyMMdd, for example, 20151111. You must specify this parameter or the startTimestampMillis and endTimestampMillis parameters. For example, Alibaba Cloud Data Process Center performs scheduling only at the day level. Therefore, the date parameter is provided.	No	None
isExportSequenceInfo	Specifies whether to export time-series information. Time-series information includes the time when data is written. The default value is *false*, indicating that time series information is not exported.	No	None
maxRetries	The maximum number of retries for each request of reading incremental data from Table Store. The default value is 30. Retries are performed at certain intervals. The total time of 30 retries is approximately 5 minutes. Generally, you can keep the default settings.	No	None
startTimeString	The left boundary of the time range (left-closed and right-open) of incremental data, measured in milliseconds in the format of `yyyymmddhh24miss`.	No	None
endTimeString	The right boundary of the time range (left-closed and right-open) of incremental data, measured in milliseconds in the format of `yyyymmddhh24miss`.	No	None
mode	The export mode. If this parameter is set to single_version_and_update_only, data is exported by row. By default, data is not exported by column.	No	None

Configure OTSStream Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for OTSStream Reader.

Configure OTSStream Reader by using the code editor

In the following code, a node is configured to export the incremental data of Table Store. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"otsstream",// The reader type.
      "parameter":{
        "statusTable":"TableStoreStreamReaderStatusTable",// The name of the table that stores st
atus records.
```

```
            "maxRetries":30,// The maximum number of retries on each request of reading incremental d
ata from Table Store. It is set to 30 by default.
            "isExportSequenceInfo":false,// Specifies whether to export the time series information.
            "datasource":"$srcDatasource",// The connection.
            "startTimeString":"${startTime}",// The start time (included) of the incremental data.
            "table":"",// The name of the table to be synchronized.
            "endTimeString":"${endTime}"// The end time (excluded) of the incremental data.
        },
        "name":"Reader",
        "category":"reader"
    },
    {
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
    ]
  }
}
```

2.4.5.3.17. Configure RDBMS Reader

This topic describes the data types and parameters supported by RDBMS Reader and how to configure it by using the code editor.

Background information

RDBMS Reader allows you to read data from an RDBMS database. RDBMS Reader connects to a remote RDBMS database and runs a SELECT statement to select and read data from the database. RDBMS Reader can read data from databases such as Dameng, Db2, PPAS, and Sybase databases. If you need RDBMS Reader to read data from a common relational database, register the driver for the corresponding database type.

RDBMS Reader connects to a remote RDBMS database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. The RDBMS database runs the statement and returns the result. Then, RDBMS Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

- RDBMS Reader generates the SQL statement based on the table, column, and where parameters that you have configured, and sends the generated SQL statement to the RDBMS database.
- If you specify the querySql parameter, RDBMS Reader directly sends the value of this parameter to the RDBMS database.

RDBMS Reader supports most data types of a common relational database, such as numbers and characters. Make sure that your data types are supported.

Parameters

Parameter	Description	Required	Default value
	The JDBC URL for connecting to the RDBMS database. The format must be in accordance with the official RDBMS specifications. You can also specify the information of the attachment facility. The format varies based on the database type. Data Integration selects an appropriate driver for data reading based on the format. • Format for DM databases: `jdbc:dm://ip:port/database` • Format for Db2 databases: `jdbc:db2://ip:port/database` • Format for PPAS databases: `jdbc:edb://ip:port/database` You can enable RDBMS Reader to support a new database by using the following method: • Go to the RDBMS Reader directory. In the directory, ${DATAX_HOME} indicates the main directory of Data Integration. • Open the *plugin.json* file in the RDBMS Reader directory, and add the driver of your database to the drivers array in the file. RDBMS Reader dynamically selects the appropriate database driver to connect to the database when nodes are run. `{ "name": "rdbmsreader", "class": "com.alibaba.datax.plugin.reader.rdbmsreader.Rdbms`		

Parameter	Description	Required	Default value
jdbcUrl	Reader", "description": "useScene: prod. mechanism: Jdbc connection using the database, execute select sql, retrieve data from the ResultSet. warn: The more you know about the database, the less problems you encounter.", "developer": "alibaba", "drivers": ["dm.jdbc.driver.DmDriver", "com.ibm.db2.jcc.DB2Driver", "com.sybase.jdbc3.jdbc.SybDriver", "com.edb.Driver"] } ``` - Add the driver package to the libs directory in the RDBMS Reader directory. ``` $tree . |-- libs | |-- Dm7JdbcDriver16.jar | |-- commons-collections-3.0.jar | |-- commons-io-2.4.jar | |-- commons-lang3-3.3.2.jar | |-- commons-math3-3.1.1.jar | |-- datax-common-0.0.1-SNAPSHOT.jar | |-- datax-service-face-1.0.23-20160120.024328-1.jar | |-- db2jcc4.jar | |-- druid-1.0.15.jar | |-- edb-jdbc16.jar | |-- fastjson-1.1.46.sec01.jar | |-- guava-r05.jar | |-- hamcrest-core-1.3.jar | |-- jconn3-1.0.0-SNAPSHOT.jar | |-- logback-classic-1.0.13.jar | |-- logback-core-1.0.13.jar | |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar | `-- slf4j-api-1.7.10.jar |-- plugin.json |-- plugin_job_template.json `-- rdbmsreader-0.0.1-SNAPSHOT.jar	Yes	None

Parameter	Description	Required	Default value
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the source table.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select specific columns to export. • The column order can be changed. You can export the specified columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in JSON format, for example, `["id","1", "'bazhen.csy'", "null", "to_char(a + 1)", "2.3" , "true"]`. ○ id: a column name. ○ 1: an integer constant. ○ 'bazhen.csy': a string constant. ○ null: a null pointer. ○ to_char(a + 1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized, and cannot be left empty.	Yes	None
splitPk	The field used for data sharding when RDBMS Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to specific shards. • The splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, RDBMS Reader returns an error. • If you do not specify the splitPk parameter or leave it empty, RDBMS Reader synchronizes data by using a single thread.	No	An empty string

Parameter	Description	Required	Default value
where	The WHERE clause. RDBMS Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, set this parameter to limit 10. To synchronize data generated on the current day, set the where parameter to `gmt_create > $bizdate` . • You can use the WHERE clause to read incremental data. • If you do not specify the where parameter or leave it empty, all data is read.	No	None
querySql	The SELECT statement used to for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you need to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id` . If you specify the querySql parameter, RDBMS Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects reading efficiency. Note A value greater than 2048 may lead to OOM during the data synchronization process.	No	*1,024*

Configure RDBMS Reader by using the codeless UI

The codeless UI is not supported for RDBMS Reader.

Configure RDBMS Reader by using the code editor

In the following code, a node is configured to read data from an RDBMS database.

```
{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  },
```

```
"setting": {
  "errorLimit": {
    "record": "0"
  },
  "speed": {
    "concurrent": 1,
    "throttle": false
  }
},
"steps": [
  {
    "category": "reader",
    "name": "Reader",
    "parameter": {
      "connection": [
        {
          "jdbcUrl": [
            "jdbc:dm://ip:port/database"
          ],
          "table": [
            "table"
          ]
        }
      ],
      "username": "username",
      "password": "password",
      "table": "table",
      "column": [
        "*"
      ],
      "preSql": [
        "delete from XXX;"
      ]
    },
    "stepType": "rdbms"
  },
  {
    "category": "writer",
    "name": "Writer",
    "parameter": {},
    "stepType": "stream"
```

```
        "stepType": "stream"
      }
    ],
    "type": "job",
    "version": "2.0"
}
```

2.4.5.3.18. Configure Stream Reader

This topic describes the data types and parameters supported by Stream Reader and how to configure it by using the code editor.

Stream Reader automatically generates data from the memory. It is mainly used for performance testing for data synchronization and basic functional testing.

The following table lists the data types supported by Stream Reader.

Data type	Description
String	A sequence of characters.
Long	A long integer.
Date	A value that represents dates.
Boolean	A Boolean data type that has one of two possible values.
Bytes	An 8-bit signed two's complement integer.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
column	The column data and type of the source data. Multiple columns can be configured. You can set to generate random strings and specify the range. The example is as follows: `"column" : [ { "random": "8,15" }, { "random": "10,10" } ]` The parameters are described as follows: • "random": "8, 15": generates a random string that is 8 to 15 bytes in length. • "random": "10, 10": generates a 10-byte random string.	Yes	None
sliceRecordCount	The number of columns generated repeatedly.	Yes	None

Configure Stream Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Stream Reader.

Configure Stream Reader by using the code editor

In the following code, a node is configured to read data from the memory and then write the data to Stream Reader.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",// The reader type.
      "parameter":{
        "column":[// The columns to be synchronized.
          {
            "type":"string",// The data type.
```

```
                "value":"field" // The value.
            },
            {
                "type":"long",
                "value":100
            },
            {
                "dateFormat":"yyyy-MM-dd HH:mm:ss",// The format of the time.
                "type":"date",
                "value":"2014-12-12 12:12:12"
            },
            {
                "type":"bool",
                "value":true
            },
            {
                "type":"bytes",
                "value":"byte string"
            }
          ],
          "sliceRecordCount":"100000"// The number of columns repeatedly generated.
        },
        "name":"Reader",
        "category":"reader"
    },
    {// The following template is used to configure the writer. For more information, see the correspo
nding topic.
        "stepType":"stream",
        "parameter":{},
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
```

```
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.3.19. Configure Hive Reader

Parameters

Parameter	Description	Required	Default value
column	The fields to read. Example: `"column": ["id", "name"]`.	Yes	None
table	The name of the Hive table to read. The name is case sensitive.	Yes	None
partition	The partition information of the table to read. The last-level partition must be specified. For example, if you want to read data from a three-level partition table, set this parameter to a value that contains the last-level partition information, such as pt=20150101/type=1/biz=2.	Yes	None

Configure Hive Reader by using the code editor

In the following code, a node is configured to read data from a Hive data store.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Writer",
      "category":"writer"
    },
    {
      "stepType":"hive", // The reader type. The name is the same as that in MaxCompute.
      "parameter":{
        "parameter":{
          "column": [// The columns to be synchronized.
            "id",
            "name"
          ],
          "table": "student_tmp_2", // The name of the table to be synchronized.
          "partition": "academy=yx/class=001", // The partition settings.
          "datasource": "hive_demo"
        }
      },
      "name":"Reader",
      "category":"reader"
    }
  ],
  "setting":{
  },
  "order":{
    "hops":[// Synchronize data from the reader to the writer.
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.3.20. Configure Elasticsearch Reader

This topic describes the working principles, features, and parameters of Elasticsearch Reader.

Working principles

- Elasticsearch Reader reads data from Elasticsearch by slicing scroll queries. The slices are processed by multiple threads of a data synchronization node.
- Data types are converted based on the mapping configuration of Elasticsearch.

Basic settings

```
{
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  },
  "setting":{
    "errorLimit":{
      "record":"0" // The maximum number of dirty data records allowed.
    },
    "jvmOption":"",
    "speed":{
      "concurrent":3,
      "throttle":false
    }
  },
  "steps":[
    {
      "category":"reader",
      "name":"Reader",
      "parameter":{
        "column":[ // The fields to read.
          "id",
          "name"
        ],
        "endpoint":"", // The endpoint.
        "index":"",  // The index name.
        "password":"",  // The password.
```

```
            "scroll":"",  // The scroll ID.
            "search":"",  // The search criteria. The value is the same as the Elasticsearch query that use
s the _search API.
            "type":"default",
            "username":""  // The username.
        },
        "stepType":"elasticsearch"
      },
      {
        "category":"writer",
        "name":"Writer",
        "parameter":{},
        "stepType":"stream"
      }
    ],
    "type":"job",
    "version":"2.0" // The version number.
}
```

Advanced features

- Supports storing all data of an Elasticsearch document in one column.

  You can create a column to store all data of an Elasticsearch document.

- Supports converting semi-structured data to structured data.

Item	Description
Background	Data in Elasticsearch is deeply nested. Elasticsearch may contain fields of various types and lengths and may use Chinese names. To facilitate data computing and storage in downstream businesses, Elasticsearch Reader supports converting semi-structured data to structured data.
Principle	Elasticsearch Reader flattens nested JSON data obtained from Elasticsearch to single-dimensional data based on the paths of properties in the JSON data. Then, Elasticsearch Reader maps the single-dimensional data to structured tables. In this way, Elasticsearch data in a complex structure is converted to multiple structured tables.

Item	Description
Solution	○ Elasticsearch Reader converts nested JSON data to single-dimensional data by using the following path formats: ■ Property ■ Property.Child property ■ Property[0].Child property ○ If a property has multiple child properties, Elasticsearch Reader traverses all data of the property and splits the data to multiple tables or multiple rows in the following format: Property[*].Child property ○ Elasticsearch Reader merges data in a string array to one property in the following format and removes duplicates: Property[] where duplicates are removed ○ Elasticsearch Reader merges multiple properties to one property in the following format: Property 1,Property 2 ○ Elasticsearch Reader presents optional properties in the following format: Property 1|Property 2

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint of Elasticsearch.	Yes	None
username	The username for HTTP authentication.	No	Empty string
password	The password for HTTP authentication.	No	Empty string
index	The index name in Elasticsearch.	Yes	None
type	The type name in the index of Elasticsearch.	No	Index name
pageSize	The number of data records to read at a time.	No	*100*
search	The query parameter of Elasticsearch.	Yes	None

Parameter	Description	Required	Default value
scroll	The scroll parameter of Elasticsearch, which sets the timestamp of the snapshot taken for a scroll.	Yes	None
sort	The field based on which the returned results are sorted.	No	None
retryCount	The number of retries after a failure.	No	*300*
connTimeOut	The connection timeout of the client.	No	*600,000*
readTimeOut	The data reading timeout of the client.	No	*600,000*
multiThread	Specifies whether to use multiple threads for an HTTP request.	No	*true*
column	The fields to read.	Yes	None
full	Specifies whether to create a column to record all data of an Elasticsearch document.	No	*false*
multi	Specifies whether to split an array to multiple rows. If you enable this feature, you need to specify additional settings.	No	*false*

Additional settings:

```
"full":false,
      "multi": {
        "multi": true,
        "key":"crn_list[*]"
      }
```

2.4.5.3.21. Configure Vertica Reader

Vertica is a column-oriented database using the Massively Parallel Processing (MPP) architecture. Vertica Reader allows you to read data from Vertica. This topic describes how Vertica Reader works, the supported parameter, and how to configure it by using the code editor.

How it works

Vertica Reader connects to a remote Vertica database by using JDBC and executes a SELECT statement to select and read data from the database.

Vertica Reader connects to a remote Vertica database by using JDBC, generates a SELECT statement based on your configurations, and then sends the statement to the database. The Vertica database executes the statement and returns the result. Then, Vertica Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and sends the datasets to a writer.

- Vertica Reader generates the SELECT statement based on the table, column, and where parameters that you have configured, and sends the generated SELECT statement to the Vertica database.
- If you specify the querySql parameter, Vertica Reader directly sends the value of this parameter to the Vertica database.

Vertica Reader accesses a Vertica database by using the Vertica database driver. Confirm the compatibility between the driver version and your Vertica database. Vertica Reader uses the following version of the Vertica database driver:

```
<dependency>
   <groupId>com.vertica</groupId>
   <artifactId>vertica-jdbc</artifactId>
   <version>7.1.2</version>
</dependency>
```

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL for connecting to the Vertica database. You can specify multiple JDBC URLs for a database. The JDBC URLs are described in a JSON array. If you specify multiple JDBC URLs, Vertica Reader verifies the connectivity of the URLs in sequence to find a valid URL. If no URL is valid, Vertica Reader returns an error. Note The jdbcUrl parameter must be included in the connection parameter. The value of the jdbcUrl parameter must be in compliance with the standard format supported by Vertica. You can also specify the information of the attachment facility. Example: `jdbc:vertica://1**.0.0.1:3306/database` .	No	None
username	The username for connecting to the Vertica database.	No	None
password	The password for connecting to the Vertica database.	No	None
table	The name of the source table from which Vertica Reader reads data. Vertica Reader can read data from multiple tables. The tables are described in a JSON array. If you specify multiple tables, make sure that the tables have the same schema. Vertica Reader does not check whether the tables have the same schema. Note The table parameter must be included in the connection parameter.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns in the source table. • Column pruning is supported. You can select specific columns to export. • The column order can be changed. You can export the specified columns in an order different from that specified in the schema of the table. • Constants are supported. • The column parameter must explicitly specify a set of columns to be synchronized, and cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when Vertica Reader reads data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to specific shards. • The splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you set this parameter to a column of an unsupported type, Vertica Reader returns an error. • If you leave the splitPk parameter empty, Vertica Reader reads data from the source table by using a single thread.	No	None
where	The WHERE clause. Vertica Reader generates a SELECT statement based on the table, column, and where parameters that you have configured, and uses the generated SELECT statement to select and read data. For example, you can specify the where parameter during testing. To synchronize data generated on the current day, set the where parameter to `gmt_create > $bizdate`. • You can use the WHERE clause to synchronize incremental data. • If you do not specify the where parameter or leave it empty, all data is read.	No	None
querySql	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. If you specify the querySql parameter, Vertica Reader ignores the table, column, and where parameters that you have configured.	No	None
fetchSize	The number of data records to read at a time. This parameter determines the number of interactions between Data Integration and the database and affects data reading efficiency. Note A value greater than 2048 may lead to OOM during the data synchronization process.	No	*1024*

Configure Vertica Reader by using the codeless UI

The codeless UI is not supported for Vertica Reader.

Configure Vertica Reader by using the code editor

In the following code, a node is configured to read data from a Vertica database.

```
{
  "type": "job",
  "steps": [
    {
      "stepType":"vertica", // The reader type.
       "parameter": {
        "datasource": "", // The connection name.
         "username": "",
        "password": "",
        "where": "",
        "column": [ // The columns to be synchronized.
            "id",
          "name"
        ],
        "splitPk": "id",
        "connection": [
          {
            "table": [ // The name of the table to be synchronized.
                "table"
            ],
            "jdbcUrl": [
              "jdbc:vertica://host:port/database"
            ]
          }
        ]
      },
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "stream",
      "parameter": {
        "print": false,
        "fieldDelimiter": ","
      },
      "name": "Writer",
      "category": "writer"
```

```
        "category": "writer"
      }
    ],
    "version": "2.0",
    "order": {
      "hops": [
        {
          "from": "Reader",
          "to": "Writer"
        }
      ]
    },
    "setting": {
      "errorLimit": {
        "record":"0" // The maximum number of dirty data records allowed.
      },
      "speed": {
        "throttle": false, // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        "concurrent": 1 // The maximum number of concurrent threads.
      }
    }
}
```

2.4.5.3.22. Configure GBase Reader

This topic describes how GBase Reader reads data and how to configure a sync node to read data from a GBase database.

GBase Reader connects to a remote GBase database through the MySQL Java Database Connectivity (JDBC) Driver, generates SQL statements based on your configurations, and then reads data from the remote GBase database. Then, GBase Reader assembles the returned data to abstract datasets in custom data types supported by Data Integration, and passes the datasets to a writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
table	The name of the table to be synchronized. You can select only one source table for each sync node.	Yes	None
column	The columns to be synchronized from the source table. The columns are described in a JSON array. The default value is [*], which indicates all columns. • Column pruning is supported. You can select and export specific columns. • Change of the column order is supported. You can export the columns in an order different from that specified in the schema of the table. • Constants are supported. The column names must be arranged in compliance with the SQL syntax supported by MySQL, for example, `["id","table","1","'mingya.wmy'","'null'","to_char(a+1)","2.3","true"]`. ○ id: a column name. ○ table: the name of a column that contains reserved keywords. ○ 1: an integer constant. ○ 'mingya.wmy': a string constant, which is enclosed in single quotation marks (' '). ○ null: ■ " " indicates an empty value. ■ null indicates a null value. ■ 'null' indicates the string null. ○ to_char(a+1): a function expression. ○ 2.3: a floating-point constant. ○ true: a Boolean value. • The column parameter must explicitly specify a set of columns to be synchronized. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
splitPk	The field used for data sharding when GBase Reader extracts data. If you specify the splitPk parameter, the table is sharded based on the shard key specified by this parameter. Data Integration then runs concurrent threads to synchronize data. This improves efficiency. • We recommend that you set the splitPk parameter to the primary key of the table. Based on the primary key, data can be well distributed to different shards, but not intensively distributed to certain shards. • Currently, the splitPk parameter supports data sharding only for integers but not for other data types such as string, floating point, and date. If you specify this parameter to a column of an unsupported type, GBase Reader ignores the splitPk parameter and synchronizes data through a single thread. • If you do not specify the splitPk parameter or leave it empty, Data Integration synchronizes data through a single thread.	No	None
where	The WHERE clause. For example, set this parameter to `gmt_create>$bizdate`. • You can use the WHERE clause to synchronize incremental data. If you do not specify the where parameter or leave it empty, all data is synchronized. • Do not set the where parameter to limit 10, which does not conform to the constraints of MySQL on the SQL WHERE clause.	No	None
querySql (only available in the code editor)	The SELECT statement used for refined data filtering. If you specify this parameter, Data Integration directly filters data based on this parameter. For example, if you want to join multiple tables for data synchronization, set this parameter to `select a,b from table_a join table_b on table_a.id = table_b.id`. The priority of the querySql parameter is higher than those of the table, column, where, and splitPk parameters. If you specify the querySql parameter, GBase Reader ignores the table, column, where, and splitPk parameters that you have configured. The datasource parameter parses information, including the username and password, from this parameter.	No	None

Configure GBase Reader by using the codeless UI

Currently, the codeless user interface (UI) is not supported for GBase Reader.

Configure GBase Reader by using the code editor

In the following code, a node is configured to read data from a GBase database.

```
{
   "type":"job",
   "version":"2.0",// The version number.
   "steps":[
      {
         "stepType":"gbase // The reader type.
         "parameter":{
            "column":[// The columns to be synchronized.
               "id"
            ],
            "connection":[
               {   "querysql":["select a,b from join1 c join join2 d on c.id = d.id;"], // Specify the querySql pa
rameter in the connection parameter as a string.
                  "datasource":"",// The connection name.
                  "table":[// The name of the table to be synchronized.
                     "xxx"
                  ]
               }
            ],
            "where":"",// The WHERE clause.
            "splitPk":"",// The shard key.
            "encoding":"UTF-8"// The encoding format.
         },
         "name":"Reader",
         "category":"reader"
      },
      {
         "stepType":"stream",
         "parameter":{},
         "name":"Writer",
         "category":"writer"
      }
   ],
   "setting":{
      "errorLimit":{
         "record":"0"// The maximum number of dirty data records allowed.
      }
```

```
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4. Configure the writer

2.4.5.4.1. Configure AnalyticDB for MySQL 2.0 Writer

This topic describes the data types and parameters supported by AnalyticDB for MySQL 2.0 Writer and how to configure it by using the codeless UI and code editor.

Prerequisites

Data Integration can import data to AnalyticDB for MySQL 2.0 in real time. This method requires you to create real-time tables, which are fact tables, in the destination AnalyticDB for MySQL 2.0 database in advance. In real-time import mode, data is imported efficiently and the process is simple.

You must configure a connection before you configure AnalyticDB for MySQL 2.0 Writer.

Data types

The following table lists the data types supported by AnalyticDB for MySQL 2.0 Writer.

Category	AnalyticDB for MySQL 2.0 data type
Integer	INT, TINYINT, SMALLINT, and BIGINT
Floating point	FLOAT and DOUBLE
String	VARCHAR
Date and time	DATE and TIMESTAMP

Category	AnalyticDB for MySQL 2.0 data type
Boolean	BOOLEAN

Parameters

Parameter	Description	Required	Default value
connectionUrl	The URL for connecting to the AnalyticDB for MySQL 2.0 database. Specify the parameter in the IP address:Port format.	Yes	None
database	The name of the AnalyticDB for MySQL 2.0 database.	Yes	None
Access Id	The AccessKey ID that you can use to connect to the AnalyticDB for MySQL 2.0 database.	Yes	None
Access Key	The AccessKey secret that you can use to connect to the AnalyticDB for MySQL 2.0 database.	Yes	None
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
partition	The partition name of the destination table. If the destination table is partitioned, this parameter is required.	No	None
writeMode	The write mode. Set the value to insert. In this mode, if a primary key conflict occurs, the conflicting rows are overwritten.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, ["a","b","c"]. Set the value to an asterisk (*) if data is written to all the columns in the destination table.	Yes	None
suffix	Optional. The suffix to the AnalyticDB for MySQL 2.0 URL that is in the format of `IP address:Port`. This suffix is a custom connection string. After this parameter is set, the URL changes to a JDBC connection string for connecting to AnalyticDB for MySQL 2.0. For example, set the suffix parameter to `autoReconnect=true&failOverReadOnly=false&maxReconnects=10`.	No	None

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. This parameter is available only when the writeMode parameter is set to insert.	Required only when the writeMode parameter is set to insert	None
bufferSize	The size of the Data Integration data buffer, which is designed to improve the performance of AnalyticDB for MySQL 2.0. Data from the source database is sorted in the buffer before the data is committed to AnalyticDB for MySQL 2.0. The data in the buffer is sorted based on the partition key columns in AnalyticDB for MySQL 2.0. In this way, the data is organized in an order that can improve the performance of the AnalyticDB for MySQL 2.0 server. Data in the buffer is committed to AnalyticDB for MySQL 2.0 in batches based on the batchSize parameter. We recommend that you set the bufferSize value to a multiple of batchSize. This parameter is available only when the writeMode parameter is set to insert.	Required only when the writeMode parameter is set to insert	Disabled

Configure AnalyticDB for MySQL 2.0 Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Write Method	The writeMode parameter in the preceding parameter description.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.

GUI element	Description
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure AnalyticDB for MySQL 2.0 Writer by using the code editor

```
{
  "type":"job",
  "version":"2.0",
  "steps":[
    {
      "stepType":"stream",
      "parameter":{
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"ads",// The writer type.
      "parameter":{
        "partition:"",// The partition name of the destination table.
        "datasource":"",// The connection name.
        "column":[// The columns to which data is written.
            "id"
        ],
        "writeMode":"insert",// The write mode.
        "batchSize":"256",// The number of data records to write at a time.
        "table":"".// The name of the destination table.
```

```
            "overWrite":"true"// Specifies whether to overwrite the destination table when data is writte
n to AnalyticDB for MySQL 2.0. A value of true indicates that the destination table is overwritten. A val
ue of false indicates that the destination table is not overwritten and the new data is appended to th
e existing data. This value takes effect only when the writeMode parameter is set to load.
        },
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4.2. Configure DataHub Writer

This topic describes the data types and parameters supported by DataHub Writer and how to configure it by using the code editor.

DataHub is a real-time data distribution platform designed to process streaming data. You can publish and subscribe applications to streaming data in DataHub and distribute the data to other platforms. This allows you to easily analyze streaming data and build applications based on the streaming data.

Based on the Apsara system of Alibaba Cloud, DataHub features high availability, low latency, high scalability, and high throughput. Seamlessly integrated with Realtime Compute, DataHub allows you to easily use SQL to analyze streaming data. DataHub can also distribute streaming data to Alibaba Cloud services such as MaxCompute and OSS.

> **Note** Strings can only be UTF-8 encoded. The size of each string must not exceed 1 MB.

Parameter configuration

The source is connected to the destination through a single channel. Therefore, the channel type configured for the writer must be the same as that configured for the reader. Generally, channels are categorized into two types: memory andfile. The following configuration sets the channel type to file:

```
"agent.sinks.dataXSinkWrapper.channel": "file"
```

Parameters

Parameter	Description	Required	Default value
accessId	The AccessKey ID for accessing DataHub.	Yes	None
accessKey	The AccessKey secret for accessing DataHub.	Yes	None
endpoint	The endpoint of DataHub.	Yes	None
maxRetryCount	The maximum number of retries if a task fails.	No	None
mode	The mode for writing strings.	Yes	None
parseContent	The data that has been parsed.	Yes	None
project	The organizational unit in DataHub. Each project contains one or more topics. **Note** DataHub projects are independent from MaxCompute projects. Projects created in MaxCompute cannot be used in DataHub.	Yes	None
topic	The minimum unit for data subscription and publication. You can use topics to distinguish different types of streaming data.	Yes	None

Parameter	Description	Required	Default value
maxCommitSize	The amount of data, in MB, that DataHub Writer buffers before sending it to the destination. This mechanism aims to improve writing efficiency. The default value is 1048576, in KB, that is, 1 MB.	No	*1048576*
batchSize	The number of data records that DataHub Writer buffers before sending them to the destination. This mechanism aims to improve writing efficiency. The default value is 1024.	No	*1,024*
maxCommitInterval	The maximum interval at which DataHub Writer sends data to the destination. When an interval ends, DataHub Writer sends buffered data even if the data amount does not reach the preceding two thresholds. The default value is 30000, in milliseconds, that is, 30 seconds.	No	*30,000*
parseMode	The mode for parsing log entries. Valid values: *default* and *csv*. The value default indicates that no log parsing is required. The value csv indicates that a delimiter is inserted between fields for each log entry.	No	*default*

Configure DataHub Writer by using the codeless UI

Currently, the codeless UI is not supported for DataHub Writer.

Configure DataHub Writer by using the code editor

In the following code, a node is configured to read data from the memory and then write the data to DataHub.

```
{
  "type": "job",
  "version": "2.0", // The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "datahub", // The writer type.
      "parameter": {
        "datasource": "", // The connection name.
```

```
            "topic": "",// The minimum unit for data subscription and publication. You can use topics to di
stinguish different types of streaming data.
            "maxRetryCount":500,// The maximum number of retries if a task fails.
            "maxCommitSize": 1048576// The amount of data, in MB, that DataHub Writer buffers before s
ending it to the destination.
            "shardId": "xxxxxx" // The shard of the DataHub topic.
        },
        "name": "Writer",
        "category": "writer"
    }
  ],
  "setting": {
    "errorLimit": {
      "record": "" // The maximum number of dirty data records allowed.
    },
    "speed": {
      "concurrent": 20, // The maximum number of concurrent threads.
      "throttle": false, // The value false indicates that the bandwidth is not throttled. The value true
indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set t
his parameter to true.
    }
  },
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
    ]
  }
}
```

2.4.5.4.3. Configure the DB2 writer

The DB2 writer enables writing data to tables stored on Db2 databases. To write data into a Db2 table, the DB2 writer connects to the remote Db2 database through JDBC, and runs `INSERT INTO` statements. Data is written into the Db2 table in batches.

The DB2 writer is designed for ETL developers to import data from data warehouses to Db2 databases. It also serves as a data migration tool for database administrators and other users.

The DB2 writer reads data from the channel, connects to a remote Db2 database through JDBC, and then runs `INSERT INTO` statements. The rows that violate the unique index constraint or primary key constraint cannot be written into the Db2 database. To improve performance, the DB2 writer makes `batch updates with the PreparedStatement method` and sets `rewriteBatchedStatements=true`. In this way, the DB2 writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

> **Note** The INSERT INTO privilege is required for data synchronization tasks with the DB2 writer. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters.

The DB2 writer supports most Db2 data types. Since still some of the Db2 data types are not supported, verify that your data types are supported.

The following table lists data types supported by the DB2 writer.

Data Integration data type	Db2 data type
Integer	SMALLINT
Floating point	DECIMAL, REAL, and DOUBLE
String	CHAR, CHARACTER, VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARCHAR, CLOB, LONG VARGRAPHIC, and DBCLOB
Date and time	DATE, TIME, and TIMESTAMP
Boolean	N/A
Binary	BLOB

Parameters

Parameter	Description	Required	Default value
jdbcUrl	The JDBC connectivity URL, used to connect to the Db2 database. In accordance with Db2 official specifications, the URL format must be jdbc:db2://ip:port/database. You can also specify the information of the attachment facility.	Yes	None
username	The username used to connect to the data source.	Yes	None
password	The password used to connect to the data source.	Yes	None
table	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: "column": ["id", "name", "age"]. Set the value to an asterisk (*) if data is written to all the columns in the destination table. Example: "column": ["*"].	Yes	None
preSql	The SQL statement runs before the data synchronization task starts. Currently, you can run only one SQL statement. For example, you can run a statement to clear outdated data.	No	None
postSql	The SQL statement runs after the data synchronization task ends. Currently, you can run only one SQL statement in wizard mode but multiple SQL statements in script mode. For example, you can run a statement to add a timestamp.	No	None
batchSize	The number of data records to write per batch. Setting this parameter can greatly reduce the interactions between Data Integration and the Db2 database over the network, and increase the throughput. However, an excessively large value may cause the running Data Integration process to become out of memory (OOM).	No	1024

Configure the DB2 writer in wizard mode

Currently, wizard mode is not supported for the DB2 writer.

Configure the DB2 writer in script mode

In the following script, a task is configured to write data to a Db2 database.

```
{
   "type":"job",
   "version":"2.0", // The version number.
   "steps":[
      { // The following template is used to configure the reader. For more information, see the corresp
onding section.
         "stepType":"stream",
         "parameter":{},
         "name":"Reader",
         "category":"reader"
      },
      {
         "stepType":"db2", // The writer type.
         "parameter":{
            "postSql":[], // The SQL statement runs after the data synchronization task ends.
            "password":"", // The password.
            "jdbcUrl":"jdbc:db2://ip:port/database", //The JDBC connectivity URL used to connect to the
```

```
            "jdbcUrl": "jdbc:db2://ip:port/database", // The JDBC connectivity URL, used to connect to the
Db2 database.
            "column":[
              "id"
            ],
            "batchSize":1024, // The number of data records to write per batch.
            "table":"", // The table name.
            "username":"", // The username.
            "preSql": [] // The SQL statement runs before the data synchronization task starts.
          },
          "name":"Writer",
          "category":"writer"
        }
    ],
    "setting":{
        "errorLimit":{
            "record":"0" // The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false, // The value false means that the bandwidth is not throttled. The value true m
eans that the bandwidth is throttled. The maximum transmission rate takes effect only if you specify t
his parameter as true.
            "concurrent":1, // The maximum number of concurrent threads.
            "dmu":1 // The number of DMUs.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.4.5.4.4. Configure DRDS Writer

This topic describes the data types and parameters supported by DRDS Writer and how to configure it by using the codeless UI and code editor.

DRDS Writer allows you to write data to tables stored in DRDS databases. DRDS Writer connects to the proxy of a remote DRDS database by using JDBC, and executes a `REPLACE INTO` statement to write data to the DRDS database.

> **Note**
> - To execute the `REPLACE INTO` statement, make sure that your table has the primary key or a unique index to avoid replicated data.
> - You must configure a connection before you configure DRDS Writer.

DRDS Writer is designed for ETL developers to import data from data warehouses to DRDS databases. DRDS Writer can also be used as a data migration tool by users such as DBAs.

DRDS Writer obtains data from a Data Integration reader, and writes the data to the destination database by executing the `REPLACE INTO` statement. If no primary key conflict or unique index conflict occurs, the action is the same as that of the `INSERT INTO` statement. If a conflict occurs, original rows are replaced by new rows. DRDS Writer sends data to the DRDS proxy when the amount of buffered data reaches a specific threshold. The proxy determines whether to write the data to one or more tables and how to route the data when it is written to multiple tables.

> **Note** A sync node that uses DRDS Writer must have at least the permission to execute the REPLACE INTO statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters.

Similar to MySQL Writer, DRDS Writer supports most MySQL data types. Make sure that your data types are supported.

The following table lists the data types supported by DRDS Writer.

Category	DRDS data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, and YEAR
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, and TIME
Boolean	BIT and BOOLEAN
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

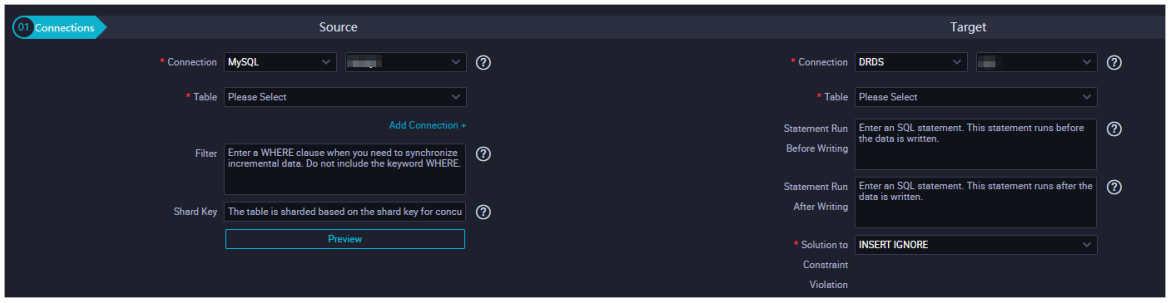
Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None

Parameter	Description	Required	Default value
table	The name of the destination table.	Yes	None
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, "column": ["id","name","age"]. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, "column":["*"].	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the DRDS database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure DRDS Writer by using the codeless UI

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Solution to Constraint Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.

Parameter	Description
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Configure DRDS Writer by using the code editor

In the following code, a node is configured to write data to a DRDS database.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
        },
    {
      "stepType":"drds",// The writer type.
      "parameter":{
        "postSql":[],// The SQL statement to execute after the sync node is run.
        "datasource":"",// The connection name.
        "column":[// The columns to which data is written.
        "id"
        ],
        "writeMode":"insert ignore",
        "batchSize":"1024",// The number of data records to write at a time.
        "table":"test",// The name of the destination table.
        "preSql":[]// The SQL statement to execute before the sync node is run.
        },
      "name":"Writer",
      "category":"writer"
        }
        ],
  "setting":{
    "errorLimit":{
```

```
        "record":"0"// The maximum number of dirty data records allowed.
          },
        "speed":{
          "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
          "concurrent":1 // The maximum number of concurrent threads.
            }
          },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
          }
        ]
      }
    }
```

2.4.5.4.5. Configure the FTP writer

The FTP writer allows you to write one or more files in CSV format into a remote FTP file. At the underlying level, this writer converts the data that is readable by the Data Integration service to CSV files, and writes these files into the remote FTP server using FTP network protocols. You must configure the data source before configuring the FTP writer.

> **Note** For more information, see Add FTP data sources.

The FTP writer can only write data into FTP files that store logical two-dimensional tables, for example, text information in the CSV format.

This writer enables you to convert data that is readable by the Data Integration service to FTP files. FTP files stores non-structured data. The advantages and disadvantages of the FTP writer are described as follows:

- Only supports text files and the schema in the text file must be a two-dimensional table. It does not support the blob type, such as video data.
- Supports CSV and text files with custom delimiters.
- Does not support text compression when data is written to the destination table.
- Supports multi-thread writing, with each thread performing write operations on a subfile.

Currently, the FTP writer does not support the following two features:

- Concurrent writing for a single file.
- Providing varying data types. The FTP does not provide data types, and the FTP writer writes data of the string type into FTP files.

Parameters

Parameter	Description	Required.	Default value
datasource	The name of the data source. You can add a data source either in wizard or script mode. The value of this parameter must be identical to the real data source name.	Yes	None
timeout	The timeout period for the connection to the FTP server, measured in milliseconds.	No	60000 (1 minute)
path	The path of the FTP file system. The write can write data into multiple files in the path.	Yes	None
FileName	The name of the file into which data is written. A random suffix is added to the file name to form the actual name of the file into which the data is written on each thread.	Yes	None
writeMode	The mode in which the FTP writer clears existing data before writing data. Valid values: • truncate: The writer clears all the files prefixed by fileName in the path before writing data. • append: No processing is performed on the file before the FTP writer imports data into this file. In the Data Integration service, the FTP writer uses the original file name in the data source. No duplicate file names are allowed. • nonConflict: An error is reported if a file prefixed by fileName exists in the path.	Yes	None
fieldDelimiter	The column delimiter of the file to be written.	Yes. A single character is used.	None
compress	The compress option. The gzip and bzip2 compression options are supported.	No	No
encoding	The encoding of the file to be read.	No	UTF-8
nullFormat	The string that represents null. Since no standard strings can represent null in text files, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat:"null"`, Data Integration considers "null" as a null pointer.	No	None
dateFormat	The date format, for example, "dateFormat": "yyyy-MM-dd".	No	None

Parameter	Description	Required.	Default value
fileFormat	The file format, including CSV and text. For the CSV format, if you want to write the data that includes column delimiters, the delimiters are escaped with quotation marks. For text format, the data to be written is separated by column delimiters without being escaped.	No	text
header	The header used when a txt file is written, for example, ['id', 'name', 'age'].	No	None
Markdonefilename	The name of the file marked as "done". After a synchronization task is completed, a MarkDoneFile is generated, based on which you can determine whether the task is executed successfully.	No	None

Configure the FTP writer in wizard mode

1. Select data sources.

   Configure the source and destination for the data synchronization task.

Parameter	Description
Data Source	The datasource parameter provided in the preceding table. Select a data source type, and enter the name of a data source that has been configured in DataWorks.
File Path	The path parameter provided in the preceding table.
Column Delimiter	The fieldDelimiter parameter provided in the preceding table. Default value: a comma (,)
Encoding	The encoding parameter provided in the preceding table. Default value: UTF-8.
Null String	The nullFormat parameter provided in the preceding table, which defines a string that represents null.
Compression Format	The nullFormat parameter provided in the preceding table. Default value: No.
Include Header	The skipHeader parameter in the preceding table. Default value: No.
Prefix Conflict	The writeMode parameter provided in the preceding table, which defines a string that represents null.

2. Configure field mappings. It is equivalent to setting the column parameter provided in the preceding table.

   You can map the left-side source table fields to the right-side destination table fields. You can also click **Add** to add a field or click the **Delete** icon to delete a field in the source table.

After you click **Map Fields in the Same Line**, each source table field is mapped to the destination table field in the same line if exists. Ensure that the conversion between data types is feasible.

3. Configure the channel.

Parameter	Description
DMU	The data processing capabilities. A data migration unit (DMU) represents the data processing capabilities for data integration, given a minimum configuration of the CPU, memory, network, and other resources.
Concurrent Jobs	The maximum number of concurrent threads to read and write data to data storage within the data synchronization task.
Transmission Rate	You can throttle the bandwidth and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Task Resource Group	The servers on which tasks are run. If an excessively large number of tasks are run on the default resource group, some tasks may be delayed due to insufficient resources. In this case, you can configure additional servers.

Configure the FTP writer in script mode

In the following script, a task is configured to write data to an FTP database.

```
{
   "type":"job",
   "version":"2.0",// The version number.
   "steps":[
      { // The following template is used to configure the reader. For more information, see the corresp
onding section.
         "stepType":"stream",
         "Parameter ":{},
         "name":"Reader",
         "category":"reader"
      },
      {
         "stepType":"ftp",// The plug-in name.
         "parameter":{
            "path":""// The file path.
            "fileName": "",// The file name.
            "nullFormat": "null", // The string that represents null.
```

```
            "dateFormat":"yyyy-MM-dd HH:mm:ss", // The time format.
            "datasource": "", // The data source.
            "writeMode": "",// The writing method.
            "fieldDelimiter": "," ,// The column delimiter.
             "encoding":"",// The encoding.
            "fileFormat": "",// The file type.
        },
        "name":"Writer",
        "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// The value false means that the bandwidth is not throttled. The value true me
ans that the bandwidth is throttled. The maximum transmission rate takes effect only if you specify th
is parameter as true.
      "concurrent":"1",// The maximum number of concurrent threads.
      "dmu":1// The number of DMUs.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4.6. Configure HBase Writer

This topic describes the features, data types, and parameters supported by HBase Writer and how to configure it by using the code editor.

HBase Writer allows you to write data to HBase data stores. Specifically, HBase Writer connects to a remote HBase data store through the Java client of HBase. Then, HBase Writer uses the PUT method to write data to the HBase data store.

Features

- HBase 0.94.x and 1.1.x are supported.
  - If you use HBase 0.94.x, set the hbaseVersion parameter to 094x for the writer.

    ```
    "writer": {
        "hbaseVersion": "094x"
      }
    ```

  - If you use HBase 1.1.x, set the hbaseVersion parameter to 11x for the writer.

    ```
    "writer": {
        "hbaseVersion": "11x"
      }
    ```

    Note Currently, HBase Writer for HBase 1.1.x is compatible with HBase 2.0. If you have any issues in using HBase Writer with HBase 2.0, submit a ticket.

- You can use concatenated fields as a rowkey.

  Currently, HBase Writer supports concatenating multiple fields to generate the rowkey of an HBase table.

- You can set the version of each HBase cell.

  The information that can be used as the version of an HBase cell includes:

  - Current time
  - Specified source column
  - Specified time

Data types

The following table lists the data types supported by HBase Writer.

Note
- The types of the specified columns must be the same as those in the HBase table.
- Data types that are not listed in the table are not supported.

Category	HBase data type
Integer	Int, Long, and Short
Floating point	Float and Double
Boolean	Boolean
String	String

Parameters

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. A value of true indicates that Kerberos authentication is required. Note • If the value is true, the following five Kerberos-related parameters must be specified: ◦ kerberosKeytabFilePath ◦ kerberosPrincipal ◦ hbaseMasterKerberosPrincipal ◦ hbaseRegionserverKerberosPrincipal ◦ hbaseRpcProtection • If the value is false, Kerberos authentication is not required and you do not need to specify the preceding parameters.	No	*false*
hbaseConfig	The properties of the HBase cluster, in JSON format. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. You can also configure other properties, such as those related to the cache and batch for scan operations.	Yes	None
mode	The mode in which data is written to the HBase data store. Currently, only the normal mode is supported. The dynamic column selection mode is coming soon.	Yes	None
table	The name of the HBase table to which data is written. The name is case-sensitive.	Yes	None
encoding	The encoding format in which a string is converted through byte[]. Currently, UTF-8 and GBK are supported.	No	*utf-8*
column	The HBase columns to which data is written. • index: the ID of the column in the source table, starting from 0. • name: the name of the column in the HBase table, in the columnFamily:column format. • type: the type of the data written, which is used by the byte[] constructor.	Yes	None

Parameter	Description	Required	Default value
maxVersion	The number of versions read by HBase Reader when multiple versions are available. Valid values: -1 and integers greater than 1. A value of -1 indicates that all versions are read.	Required in multiVersionFixedColumn mode	None
range	The rowkey range that HBase Reader reads. • startRowkey: the start rowkey. • endRowkey: the end rowkey. • isBinaryRowkey: the operation called by byte[] to convert the specified start and end rowkeys. Default value: false. If the value is true, Bytes.toBytesBinary(rowkey) is called. If the value is false, Bytes.toBytes(rowkey) is called. Example: `"range": { "startRowkey": "aaa", "endRowkey": "ccc", "isBinaryRowkey":false }` Example: `"column": [ { "index":1, "name": "cf1:q1", "type": "string" }, { "index":2, "name": "cf1:q2", "type": "string" } ]`	No	None

Parameter	Description	Required	Default value
rowkeyColumn	The rowkey of each HBase cell. • index: the ID of the column in the source table, starting from 0. If the column is a constant, set the value to -1. • type: the type of the data written, which is used by the byte[] constructor. • value: a constant, which is usually used as the delimiter between fields. HBase Writer sequentially concatenates all columns specified in this parameter to a string, and uses the string as the rowkey. The specified columns cannot be all constants. Example: `"rowkeyColumn": [ { "index":0, "type":"string" }, { "index":-1, "type":"string", "value":"_" } ]`	Yes	None

Parameter	Description	Required	Default value
versionColumn	The version of each HBase cell. You can use the current time, a specified source column, or a specified time as the version. If you do not specify this parameter, the current time is used. • index: the ID of the column in the source table, starting from 0. Make sure that the value can be properly converted to the Long type. • type: the data type. If the type is Date, HBase Writer converts the date to yyyy-MM-dd HH:mm:ss or yyyy-MM-dd HH:mm:ss SSS. If you want to use a specified time as the version, set the value to -1. • value: the specified time of the Long type. Example: "versionColumn":{ "index":1 } "versionColumn":{ "index": - 1, "value":123456789 }	No	None
nullMode	The method of processing null values. Valid values: • skip: HBase Writer does not write null values to the HBase data store. • empty: HBase Writer writes HConstants.EMPTY_BYTE_ARRAY (new byte [0]) to the HBase data store instead of null values.	No	skip
walFlag	Specifies whether to enable write ahead logging (WAL) for HBase. If the value is true, all edits requested by an HBase client for all Regions carried by the RegionServer are recorded first in the WAL (that is, the HLog). After the edits are successfully recorded in the WAL, they are implemented to the Memstore and a success indication is sent to the HBase client. If edits fail to be recorded in the WAL, a failure indication is sent to the HBase client without implementing the edits. If the value is false, WAL is disabled but writing efficiency is improved.	No	*false*

Parameter	Description	Required	Default value
writeBufferSize	The write buffer size, in bytes, of the HBase client. If you specify this parameter, you must also specify the autoflush parameter. autoflush: • If the value is true, the HBase client sends a PUT request each time it receives an edit. • If the value is false, the HBase client sends a PUT request only when its write buffer is full.	No	8 MB

Configure HBase Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for HBase Writer.

Configure HBase Writer by using the code editor

In the following code, a node is configured to write data to an HBase 1.1.x data store.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"hbase",// The writer type.
      "parameter":{
        "mode":"normal",// The mode in which data is written to the HBase data store.
        "walFlag":"false",// WAL is disabled for HBase.
        "hbaseVersion":"094x",// The HBase version.
        "rowkeyColumn":[// The rowkey of each HBase cell.
          {
            "index":"0",// The ID of the column in the source table.
            "type":"string"// The data type.
          },
          {
            "index":"-1",
            "type":"string",
            "value":" "
```

```
                "value": "_
            }
          ],
          "nullMode":"skip",// The method of processing null values.
          "column":[// The HBase columns to which data is written.
            {
              "name":"columnFamilyName1:columnName1",// The name of the HBase column.
              "index":"0",// The ID of the column in the source table.
              "type":"string"// The data type.
            },
            {
              "name":"columnFamilyName2:columnName2",
              "index":"1",
              "type":"string"
            },
            {
              "name":"columnFamilyName3:columnName3",
              "index":"2",
              "type":"string"
            }
          ],
          "writeMode":"api",// The write mode.
          "encoding":"utf-8",// The encoding format.
          "table":"",// The name of the destination table.
          "hbaseConfig":{// The properties of the HBase cluster, in JSON format.
            "hbase.zookeeper.quorum":"hostname",
            "hbase.rootdir":"hdfs: //ip:port/database",
            "hbase.cluster.distributed":"true"
          }
        },
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
```

```
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.4.7. Configure HBase11xsql Writer

This topic describes the features, data types, and parameters supported by HBase11xsql Writer and how to configure it by using the code editor.

Background information

HBase11xsql Writer allows you to write data in batches to HBase tables created by using Phoenix. Phoenix can encode the primary key to rowkey. If you directly use the HBase API to write data to an HBase table that is created by using Phoenix, you must manually convert data, which is troublesome and error-prone. HBase11xsql Writer allows you to write data to HBase tables that packs all values into a single cell per column family.

HBase11xsql Writer connects to a remote HBase data store by using JDBC, and executes an UPSERT statement to write data to the HBase data store.

Limits

- The column order specified in the writer must match that specified in the reader. When you configure the column order in the reader, you specify the order of columns in each row for the output data. When you configure the column order in the writer, you specify the expected order of columns for the input data. Example:

  Column order specified in the reader: c1, c2, c3, c4.

  Column order specified in the writer: x1, x2, x3, x4.

  In this case, the value of column c1 is assigned to column x1 in the writer. If the column order specified in the writer is x1, x2, x4, x3, the value of column c3 is assigned to column x4 and the value of column c4 is assigned to column x3.
- HBase11xsql Writer can write data only to HBase 1.x.
- HBase11xsql Writer can write data only to tables created by using Phoenix but not native HBase tables.
- HBase11xsql Writer cannot write data with timestamps.

Features

HBase11xsql Writer can write data of an indexed table and synchronously update all indexed tables.

How it works

HBase11xsql Writer connects to an HBase data store by using Phoenix, which is a JDBC driver, and executes an UPSERT statement to write data in batches to the destination table. Phoenix allows to synchronously update indexed tables when you write data.

Parameters

Parameter	Description	Required	Default value
plugin	The writer type. Set this value to hbase11xsql.	Yes	None
table	The name of the destination table. The name is case-sensitive. Generally, the name of a table that is created by using Phoenix consists of uppercase letters.	Yes	None
column	The name of the column. The name is case-sensitive. Generally, the name of each column in a table that is created by using Phoenix consists of uppercase letters. Note • HBase11xsql Writer writes data strictly in accordance with the order of the columns obtained from the reader. • You do not need to specify the data type for each column. HBase11xsql Writer automatically obtains the metadata of columns from Phoenix.	Yes	None
hbaseConfig	The properties of the HBase cluster. The hbase.zookeeper.quorum parameter is required. It specifies the ZooKeeper ensemble servers. Note • Separate the IP addresses with commas (,), for example, ip1,ip2,ip3. • The zookeeper.znode.parent parameter is optional. Default value: /hbase.	Yes	None
batchSize	The number of data records to write at a time.	No	*256*

Parameter	Description	Required	Default value
nullMode	The method of processing null values. Valid values: • *skip*: HBase11xsql Writer does not write null values to the HBase data store. • *empty*: HBase11xsql Writer writes 0 or an empty string instead of null values to the HBase data store. For a column of the numeric type, HBase11xsql Writer writes 0. For a column of the VARCHAR type, HBase11xsql Writer writes an empty string.	No	*skip*

Configure HBase11xsql Writer by using the code editor

In the following code, a node is configured to write data to an HBasae database.

```
{
 "type": "job",
 "version": "1.0",
 "configuration": {
  "setting": {
   "errorLimit": {
    "record": "0"
   },
   "speed": {
    "mbps": "1",
    "concurrent": "1"
   }
  },
  "reader": {
   "plugin": "odps",
   "parameter": {
    "datasource": "",
    "table": "",
    "column": [],
    "partition": ""
   }
  },
  "plugin": "hbase11xsql",
  "parameter": {
   "table": "The case-sensitive name of the destination table",
   "hbaseConfig": {
    "hbase.zookeeper.quorum": "The IP addresses of ZooKeeper ensemble servers of the destination HBase cluster. Obtain the IP addresses from product engineers (PEs).",
    "zookeeper.znode.parent": "The root znode of the destination HBase cluster. Obtain the IP addresses from PEs."
   },
   "column": [
    "columnName"
   ],
   "batchSize": 256,
   "nullMode": "skip"
  }
 }
}
```

FAQ

Q: What is the proper number of concurrent threads? Can I increase the number of concurrent threads to speed up the synchronization?

A: In the data import process, the default size of a JVM heap is 2 GB. Concurrent synchronization requires multiple threads. However, excessive threads sometimes cannot speed up the synchronization and may even deteriorate the performance because of frequent garbage collection (GC). We recommend that you set the number of concurrent threads within the range from 5 to 10.

Q: What is the proper value for the batchSize parameter?

A: The default value of the batchSize parameter is 256. You can set a proper value for the batchSize parameter based on the data volume of each row. Generally, the data volume of each write operation is 2 MB to 4 MB. You can set the value to the data volume of a write operation divided by the data volume of a row.

2.4.5.4.8. Configure HDFS Writer

This topic describes the data types and parameters supported by HDFS Writer and how to configure it by using the code editor.

HDFS Writer allows you to write text, ORC, or Parquet files to the specified directory in HDFS. In addition, you can associate the fields in the files with those in Hive tables. You must configure a connection before you configure HDFS Writer.

How it works

HDFS Writer writes files to HDFS in the following way:

1. Creates a temporary directory that does not exist in HDFS based on the path parameter you specified.

   The name of the temporary directory is in the format of path_Random suffix.
2. Writes files that are read by a Data Integration reader to the temporary directory.
3. Moves the files from the temporary directory to the specified directory in HDFS after all the files are written. HDFS Writer ensures that the file names do not conflict with existing files in HDFS when it moves the files.
4. Deletes the temporary directory. If the deletion is interrupted because HDFS Writer fails to connect to HDFS, you must manually delete the temporary directory.

> **Note** To synchronize data, use an administrator account with the read and write permissions.

Limits

- HDFS Writer can write only text, ORC, and Parquet files that store logical two-dimensional tables to HDFS.
- HDFS is a distributed file system and does not have a schema. Therefore, you cannot write only some of the columns in a file to HDFS.
- HDFS Writer supports only the following Hive data types:
  - Numeric: TINYINT, SMALLINT, INT, BIGINT, FLOAT, and DOUBLE

  - String: STRING, VARCHAR, and CHAR
  - Boolean: BOOLEAN
  - Date and time: DATE and TIMESTAMP
- HDFS Writer does not support other Hive data types, such as DECIMAL, BINARY, ARRAY, MAP, STRUCT, or UNION.
- HDFS Writer can write data to only one partition in a partitioned Hive table at a time.
- To write a text file to HDFS, make sure that the delimiter in the file is the same as that in the Hive table to be associated with the file. Otherwise, you cannot associate the fields in the file stored in HDFS with those in the Hive table.
- HDFS Writer can be used in the environment where Hive 1.1.1 and Hadoop 2.7.1 (JDK version: 1.7) are installed. HDFS Writer can write files to HDFS properly in testing environments where Hadoop 2.5.0, Hadoop 2.6.0, or Hive 1.2.0 is installed.

Data types

HDFS Writer supports most Hive data types. Make sure that your data types are supported.

The following table lists the Hive data types supported by HDFS Writer.

> **Note** The types of the specified columns must be the same as those of columns in the Hive table.

Category	Hive data type
Integer	TINYINT, SMALLINT, INT, and BIGINT
Floating point	FLOAT and DOUBLE
String	CHAR, VARCHAR, and STRING
Boolean	BOOLEAN
Date and time	DATE and TIMESTAMP

Parameters

Parameter	Description	Required	Default value
defaultFS	The address of the HDFS NameNode, for example, `hdfs://127.0.0.1:9000`. The default resource group does not support configuring advanced Hadoop parameters related to the high availability feature.	Yes	None

Parameter	Description	Required	Default value
fileType	The format of the files to be written to HDFS. Valid values: • *text*: the text file format. • *orc*: the ORC file format. • *parquet*: the common Parquet file format.	Yes	None
path	The directory in HDFS to which the files are written. HDFS Writer concurrently writes multiple files to the directory based on the concurrency setting. To associate the fields in a file with those in a Hive table, set the path parameter to the storage path of the Hive table in HDFS. Assume that the storage path specified for the data warehouse of Hive is `/user/hive/warehouse/`. The storage path of the hello table created in the test database is `/user/hive/warehouse/test.db/hello`.	Yes	None
fileName	The name prefix of the files to be written to HDFS. A random suffix is appended to the specified prefix to form the actual file name used by each thread.	Yes	None

Parameter	Description	Required	Default value
column	The columns to be written to HDFS. You cannot write only some of the columns in a file to HDFS. To associate the fields in a file with those in a Hive table, specify the name and type parameters for each field. You can also specify the column parameter in the following way: `"column": [ { "name": "userName", "type": "string" }, { "name": "age", "type": "long" } ]`	Yes (Not required if the fileType parameter is set to parquet)	None
writeMode	The mode in which HDFS Writer writes the files. Valid values: • *append*: writes the files based on the specified file name prefix and ensures that the actual file names do not conflict with those of existing files. • *nonConflict*: returns an error if a file with the specified file name prefix exists in the destination directory. Note Parquet files do not support the append mode. They support only the nonConflict mode.	Yes	None
fieldDelimiter	The column delimiter used in the files to be written to HDFS. Make sure that you use the same delimiter as that in the Hive table. Otherwise, you cannot query data in the Hive table.	Yes (Not required if the fileType parameter is set to parquet)	None

Parameter	Description	Required	Default value
compress	The compression format of the files to be written to HDFS. By default, this parameter is left empty, that is, files are not compressed. For a text file, the GZIP and BZIP2 compression formats are supported. For an ORC file, the SNAPPY compression format is supported. To compress an ORC file, you must install SnappyCodec.	No	None
encoding	The encoding format of the files to be written to HDFS.	No	None

Parameter	Description	Required	Default value
parquetSchema	The schema of the files to be written to HDFS. This parameter is required only when the fileType parameter is set to parquet. Format: `message messageTypeName {` `required, dataType, columnName;` `..... ;` `}` Parameter description: • messageTypeName: the name of the MessageType object. • required: specifies whether the field is required. We recommend that you set the parameter to optional for all fields. • dataType: the type of the field. Valid values: BOOLEAN, INT32, INT64, INT96, FLOAT, DOUBLE, BINARY, and FIXED_LEN_BYTE_ARRAY. Set this parameter to BINARY if the field stores strings. Note Each line, including the last one, must end with a semicolon (;). Example: `message m {` `optional int64 id;` `optional int64 date_id;` `optional binary datetimestring;` `optional int32 dspId;` `optional int32 advertiserId;` `optional int32 status;` `optional int64 bidding_req_num;` `optional int64 imp;` `optional int64 click_num;` `}`	No	None

Parameter	Description	Required	Default value
hadoopConfig	The advanced parameter settings of Hadoop, such as those related to high availability. The default resource group does not support configuring advanced Hadoop parameters related to the high availability feature. "hadoopConfig":{ "dfs.nameservices": "testDfs", "dfs.ha.namenodes.testDfs": "namenode1,namenode2", "dfs.namenode.rpc-address.youkuDfs.namenode1": "", "dfs.namenode.rpc-address.youkuDfs.namenode2": "", "dfs.client.failover.proxy.provider.testDfs": "org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider" }	No	None
	The synchronization mode for Parquet files. If the dataxParquetMode parameter is set to fields, you can write data of complex types, such as ARRAY, MAP, and STRUCT. Valid values: fields and columns. If the dataxParquetMode parameter is set to fields, HDFS Writer supports HDFS over OSS. HDFS uses OSS as the storage service and HDFS Writer writes Parquet files to OSS. In this case, you can add the following OSS-related parameters in the hadoopConfig parameter: • fs.oss.accessKeyId: the AccessKey ID for connecting to OSS. • fs.oss.accessKeySecret: the AccessKey secret for connecting to OSS. • fs.oss.endpoint: the endpoint for connecting to OSS. Example:		

Parameter	Description	Required	Default value
dataxParquetMode	```json "writer": { "name": "hdfswriter", "parameter": { "defaultFS": "oss://test-bucket", "fileType": "parquet", "path": "/datasets/oss_demo/kpt", "fileName": "test", "writeMode": "truncate", "compress": "SNAPPY", "encoding": "UTF-8", "hadoopConfig": { "fs.oss.accessKeyId": "the-access-id", "fs.oss.accessKeySecret": "the-access-key", "fs.oss.endpoint": "oss-cn-hangzhou.aliyuncs.com" }, "parquetSchema": "message test {\n required int64 id;\n optional binary name (UTF8);\n optional int64 gmt_create;\n required group map_col (MAP) {\n repeated group key_value {\n required binary key (UTF8);\n required binary value (UTF8);\n }\n }\n required group array_col (LIST) {\n repeated group list {\n required binary element (UTF8);\n }\n }\n required group struct_col {\n required int64 id;\n required binary name (UTF8);\n } \n}", "dataxParquetMode": "fields" } } ```	No	*columns*

Parameter	Description	Required	Default value
haveKerberos	Specifies whether Kerberos authentication is required. Default value: *false*.	If you set this parameter to *true*, you must also set the kerberosKeytabFilePath and kerberosPrincipal parameters.	*false*
kerberosKeytabFilePath	The absolute path of the keytab file for Kerberos authentication.	Required if the haveKerberos parameter is set to true	None
kerberosPrincipal	The Kerberos principal to which Kerberos can assign tickets. Example: ****/hadoopclient@**.***. Note The absolute path of the keytab file is required for Kerberos authentication. Therefore, you can configure Kerberos authentication only on a custom resource group. Example: "haveKerberos":true, "kerberosKeytabFilePath":"/opt/datax/**.keytab", "kerberosPrincipal":"**/hadoopclient@**.**"	Required if the haveKerberos parameter is set to true	None

Configure HDFS Writer by using the codeless UI

The codeless UI is not supported for HDFS Writer.

Configure HDFS Writer by using the code editor

In the following code, a node is configured to write files to HDFS. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "version": "2.0",// The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
```

```
        "name": "Reader",
        "category": "reader"
      },
      {
        "stepType": "hdfs",// The writer type.
        "parameter": {
          "path": "",// The directory in HDFS to which the files are written.
          "fileName": "",// The name prefix of the files to be written to HDFS.
          "compress": "",// The compression format of the files.
          "datasource": "",// The connection name.
          "column": [
            {
              "name": "col1",// The name of the column.
              "type": "string"// The data type of the column.
            },
            {
              "name": "col2",
              "type": "int"
            },
            {
              "name": "col3",
              "type": "double"
            },
            {
              "name": "col4",
              "type": "boolean"
            },
            {
              "name": "col5",
              "type": "date"
            }
          ],
          "writeMode": "",// The write mode.
          "fieldDelimiter": ",",// The column delimiter.
          "encoding": "",// The encoding format.
          "fileType": "text"// The file format.
        },
        "name": "Writer",
        "category": "writer"
      }
    ],
```

```
    "setting": {
      "errorLimit": {
        "record": ""// The maximum number of dirty data records allowed.
      },
      "speed": {
        "concurrent": 3,// The maximum number of concurrent threads.
        "throttle": false // Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
      }
    },
    "order": {
      "hops": [
        {
          "from": "Reader",
          "to": "Writer"
        }
      ]
    }
}
```

2.4.5.4.9. Configure MaxCompute Writer

This topic describes the data types and parameters supported by MaxCompute Writer and how to configure it by using the codeless UI and code editor.

MaxCompute Writer is designed for developers to insert data to or update data in MaxCompute. MaxCompute Writer is suitable for importing data at the GB or TB level to MaxCompute.

> **Note** You must configure a connection before you configure MaxCompute Writer.

Based on the specified information such as the source project, table, partition, and field, MaxCompute Writer writes data to MaxCompute by using Tunnel.

Data types

The following table lists the data types supported by MaxCompute Writer.

Category	MaxCompute data type
Integer	BIGINT
Floating point	DOUBLE and DECIMAL
String	STRING

Category	MaxCompute data type
Date and time	DATETIME
Boolean	BOOLEAN

Parameters

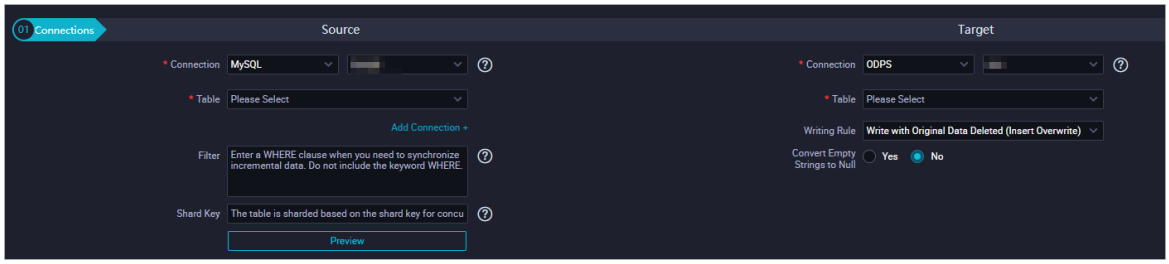
Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table. The name is not case-sensitive. You can specify only one table as the destination table.	Yes	None
partition	The partition to which data is written. The last-level partition must be specified. For example, if you want to write data to a three-level partitioned table, set the partition parameter to a value that contains the third-level partition information, for example, `pt=20150101, type=1, biz=2`. • To write data to a non-partitioned table, do not set this parameter. The data is directly written to the destination table. • MaxCompute Writer does not support writing data based on the partition route. To write data to a partitioned table, make sure that data is written to the last-level partition.	Required only for writing data to a partitioned table	None
column	The columns in the destination table to which data is written. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`. Set the value to the specified columns if data is written to only some of the columns in the destination table. Separate the columns with commas (,). Example: `"column":["id","name"]`. • MaxCompute Writer can filter columns and change the order of columns. For example, a MaxCompute table has three columns: a, b, and c. If you want to write data only to column c and column b, you can set the column parameter in the format `"column":["c","b"]`. During data synchronization, column a is automatically set to null. • The column parameter must explicitly specify a set of columns to which data is written. The parameter cannot be left empty.	Yes	None

Parameter	Description	Required	Default value
truncate	To ensure the idempotence of write operations, set the truncate parameter in the format `"truncate":"true"`. When a failed sync node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it imports the source data again. This ensure that the same data is written for each rerun. MaxCompute Writer uses MaxCompute SQL to delete data. MaxCompute SQL cannot ensure the atomicity. Therefore, the truncation operation is not an atomic operation. Conflicts may occur when concurrent nodes delete data from the same table or partition. To avoid this issue, we recommend that you do not run concurrent DDL nodes to write data to the same partition. You can create different partitions for nodes that need to run concurrently.	Yes	None

Configure MaxCompute Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.



Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.

Parameter	Description
Partition Key Column	To write data to all the columns in the destination table, set "column":["*"] for the column parameter in the preceding parameter description. The partition parameter allows you to use wildcards and specify one or more partitions. ○ `"partition":"pt=20140501/ds=*"` specifies that data is written to all ds partitions with pt=20140501. ○ `"partition":"pt=top?"` specifies that data is written to the partitions with pt=top and pt=to. You can specify the partition key columns to which data is written. Assume that the partition key column of a MaxCompute table is pt=${bdp.system.bizdate}. You can configure the column to which data is written to pt. Ignore it if the column is marked as unidentified. ○ To write data to all partitions, enter pt=*. ○ To write data to some of the partitions, specify the corresponding dates.
Writing Rule	○ **Write with Original Data Deleted (Insert Overwrite)**: All data in the table or partition is deleted before data import. This rule is equivalent to the `INSERT OVERWRITE` statement. ○ **Write with Original Data Retained (Insert Into)**: No data is deleted before data import. New data is always appended upon each run. This rule is equivalent to the `INSERT INTO` statement. **Note** ○ MaxCompute Reader reads data by using Tunnel. Sync nodes do not support data filtering. Instead, they must read all the data in a specific table or partition. ○ MaxCompute Writer writes data by using Tunnel instead of the INSERT INTO statement. You can view the complete data in the destination table only after a sync node is run. Pay attention to the node dependencies.
Convert Empty Strings to Null	Specifies whether to convert empty strings to null. Default value: *No*.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MaxCompute Writer by using the code editor

In the following code, a node is configured to write data to a MaxCompute project. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
```

```
      "category":"reader"
    },
    {
      "stepType":"odps",// The writer type.
      "parameter":{
        "partition":"",// The partition information.
        "truncate":true,// The write rule.
        "compress":false,// Specifies whether to enable compression.
        "datasource":"odps_first",// The connection name.
      "column":[// The columns to which data is written.
        "id",
        "name",
        "age",
        "sex",
        "salary",
        "interest"
        ],
        "emptyAsNull":false,// Specifies whether to convert empty strings to null.
        "table":""// The name of the destination table.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1 // The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
```

```
      ]
    }
}
```

Additional instructions

- Column filter

  By configuring MaxCompute Writer, you can perform operations that MaxCompute does not support, for example, filter columns, reorder columns, and set empty fields to null. To write data to all the columns in the destination table, set the column parameter in the format `"column":["*"]`.

  For example, a MaxCompute table has three columns: a, b, and c. If you want to write data only to column c and column b, you can set the column parameter in the format `"column":["c","b"]`. The first column and the second column of the source data are written to column c and column b in the MaxCompute table respectively. During data synchronization, column a is automatically set to null.

- Column configuration error handling

  To avoid losing the data of redundant columns and ensure high data reliability, MaxCompute Writer returns an error message if the number of columns to be written is more than that in the destination table. For example, if a MaxCompute table contains columns a, b, and c, MaxCompute Writer returns an error message if more than three columns are to be written to the table.

- Partition configuration

  MaxCompute Writer can write data only to the last-level partition, and cannot write data to the specified partition based on a field. To write data to a partitioned table, specify the last-level partition. For example, if you want to write data to a three-level partitioned table, set the partition parameter to a value that contains the third-level partition information, for example, `pt=20150101, type=1, biz=2`. The data cannot be written if you set the partition parameter to `pt=20150101, type=1` or `pt=20150101`.

- Node rerunning

  To ensure the idempotence of write operations, set the `truncate` parameter to true. When a failed sync node is rerun due to a write failure, MaxCompute Writer deletes the data that has been written before it imports the source data again. This ensures that the same data is written for each rerun. If a sync node is interrupted due to other exceptions, the data cannot be rolled back and the node cannot be rerun automatically. You can ensure the idempotence of write operations and the data integrity By setting the truncate parameter to true.

  > **Note** If the truncate parameter is set to true, all data of the specified partition or table is deleted before a rerun. Exercise caution when you set this parameter to true.

2.4.5.4.10. Configure Memcache Writer

This topic describes the data types and parameters supported by Memcache Writer and how to configure it by using the code editor.

ApsaraDB for Memcache is a distributed in-memory database service with high performance, reliability, and scalability. Based on the Apsara distributed operating system and high-performance storage technologies, ApsaraDB for Memcache provides a complete database solution with hot standby, fault recovery, business monitoring, and data migration features.

ApsaraDB for Memcache is immediately available after an instance is created. It relieves the load on databases from dynamic websites and applications by caching data in the memory and therefore improves the response speed of websites and applications.

Same as on-premises Memcached databases, ApsaraDB for Memcache databases are compatible with the Memcached protocol. ApsaraDB for Memcache databases can be directly used in your environments. The difference is that the data, hardware infrastructure, network security, and system maintenance services used by ApsaraDB for Memcache databases are all deployed in the cloud. These services are billed based on the pay-as-you-go billing method.

Memcache Writer writes data to ApsaraDB for Memcache databases based on the Memcached protocol.

Memcache Writer writes data only in text format. The method of converting data types varies based on the format of writing data.

- text: Memcache Writer uses the specified column delimiter to serialize source data to a string.
- binary: This format is not supported.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
writeMode	The write mode. Valid values: • *set*: stores the source data. • *add*: stores the source data only when its key does not exist in the destination ApsaraDB for Memcache database. This mode is not supported now. • *replace*: uses the source data to replace the data record with the same key in the destination ApsaraDB for Memcache database. This mode is not supported now. • *append*: adds the value of the source data to the end of the value of an existing data record with the same key in the destination ApsaraDB for Memcache database, but does not update the expiration time of the existing data record. This mode is not supported now. • *prepend*: adds the value of the source data to the beginning of the value of an existing data record with the same key in the destination ApsaraDB for Memcache database, but does not update the expiration time of the existing data record. This mode is not supported now.	Yes	None

Parameter	Description	Required	Default value
writeFormat	The format in which Memcache Writer writes the source data. Currently, only the text format is supported. text: serializes the source data to the text format. Memcache Writer uses the first column of the source data as the key and serializes the subsequent columns to the value by using the specified delimiter. Then, Memcache Writer writes the key-value pair to ApsaraDB for Memcache. Assume that the following source data exists: `| ID | NAME | COUNT|` `| ---- |:------|:-----|` `| 23 | "CDP" | 100 |` If you set the column delimiter to a backslash and a caret (\^), data is written to ApsaraDB for Memcache in the following format: `| KEY (OCS) | VALUE(OCS) |` `| --------- |:---------- |` `| 23 | CDP\^100 |`	No	None
expireTime	The expiration time of the source data to be cached in ApsaraDB for Memcache. ApsaraDB for Memcache supports the following two types of expiration time: • unixtime: the UNIX timestamp, indicating a specific time point in the future when the data expires. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970. • seconds: the relative time in seconds starting from the current time point. It specifies the period during which data is valid. Note If the specified time exceeds 30 days, the server identifies the time as the UNIX timestamp.	No	*0*, indicating that the data never expires
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the ApsaraDB for Memcache database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Memcache Writer by using the codeless UI

The codeless UI is not supported for Memcache Writer.

Configure Memcache Writer by using the code editor

In the following code, a node is configured to write data to an ApsaraDB for Memcache database.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"ocs",// The writer type.
      "parameter":{
        "writeFormat":"text",// The format in which Memcache Writer writes the source data.
        "expireTime":1000,// The expiration time of the source data to be cached in ApsaraDB for Memcache.
        "indexes":0,
        "datasource":"",// The connection name.
        "writeMode":"set",// The write mode.
        "batchSize":"256"// The number of data records to write at a time.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0" // The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1 // The maximum number of concurrent threads.
    }
  },
  "order":{
```

```
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4.11. Configure MongoDB Writer

This topic describes the data types and parameters supported by MongoDB Writer and how to configure it by using the code editor.

MongoDB Writer connects to a remote MongoDB database by using the Java client named MongoClient and writes data to the database. The latest version of MongoDB has improved the locking feature from database locks to document locks. The powerful index functionalities of MongoDB enable MongoDB Writer to efficiently write data to MongoDB databases. If you want to update data, specify the primary key.

> Note
> - You must configure a connection before you configure MongoDB Writer.
> - If you use ApsaraDB for MongoDB, the MongoDB database has a root account by default.
> - For security concerns, Data Integration only supports access to a MongoDB database by using a MongoDB database account. When you add a MongoDB connection, do not use the root account for access.

MongoDB Writer obtains data from a Data Integration reader, and converts the data types to those supported by MongoDB. Data Integration does not support arrays. MongoDB supports arrays and the array index is useful.

To use MongoDB arrays, you can convert strings to MongoDB arrays by configuring a parameter and write the arrays to a MongoDB database.

Data types

MongoDB Writer supports most MongoDB data types. Make sure that your data types are supported.

The following table lists the data types supported by MongoDB Writer.

Category	MongoDB data type
Integer	INT and LONG
Floating point	DOUBLE
String	STRING and ARRAY

Category	MongoDB data type
Date and time	DATE
Boolean	BOOL
Binary	BYTES

Note When data of the DATE type is written to a MongoDB database, the type of the data is converted to DATETIME.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
collectionName	The name of the MongoDB collection.	Yes	None
column	The columns in MongoDB. • name: the name of the column. • type: the data type of the column. • splitter: the delimiter. Specify this field only when you want to convert the string to an array. The string is split based on the specified delimiter, and the split strings are saved in a MongoDB array.	Yes	None
writeMode	Specifies whether to overwrite data. • isReplace: If you set this parameter to true, MongoDB Writer overwrites the data in the destination table with the same primary key. If you set this parameter to false, the data is not overwritten. • replaceKey: the primary key for each record. Data is overwritten based on this primary key. The primary key must be unique.	No	None
	The action to perform before the sync node is run. For example, you can clear outdated data before data synchronization. If the preSql parameter is left empty, no action is performed before data synchronization. Make sure that the value of the preSql parameter complies with the JSON syntax. The format requirements for the preSql parameter are as follows: • Configure the type field to specify the action type. Valid values: drop and remove. Example: `"preSql":{"type":"remove"}` .		

Parameter	Description	Required	Default value
preSql	○ *drop*: deletes the collection specified by the collectionName parameter and the data in the collection. ○ *remove*: deletes data based on conditions. ○ *json*: the conditions for deleting data. Example: `"preSql":{"type":"remove", "json":"{'operationTime':{'$gte':ISODate('${last_day}T00:00:00.424+0800')}}"}`. In the preceding JSON string, `${last_day}` is a scheduling parameter of DataWorks. The format is `$[yyyy-mm-dd]`. You can use comparison operators (such as $gt, $lt, $gte, and $lte), logical operators (such as $and and $or), and functions (such as max, min, sum, avg, and ISODate) supported by MongoDB as needed. For more information, see the MongoDB query syntax. Data Integration uses the following standard MongoDB API to query and delete the specified data: `query=(BasicDBObject) com.mongodb.util.JSON.parse(json);` `col.deleteMany(query);` Note If you want to delete data based on conditions, we recommend that you specify the conditions in JSON format preferentially. ○ *item*: the name, condition, and value for filtering data. Example: `"preSql":{"type":"remove","item":[{"name":"pv","value":"100","condition":"$gt"},{"name":"pid","value":"10"}]}`. Data Integration sets query conditions based on the value of the item field and deletes data by using the standard MongoDB API. Example: `col.deleteMany(query);`. • If the value of the preSql parameter cannot be recognized, no action is performed.	No	None

Parameter	Description	Required	Default value

Configure MongoDB Writer by using the codeless UI

The codeless UI is not supported for MongoDB Writer.

Configure MongoDB Writer by using the code editor

In the following code, a node is configured to write data to a MongoDB database. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "version": "2.0",// The version number.
  "steps": [
    {
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "mongodb",// The writer type.
      "parameter": {
        "datasource": "",// The connection name.
        "column": [
          {
            "name": "_id",// The name of the column to which data is written.
            "type": "ObjectId"// The data type of the column to which data is written. If the replaceke
y parameter is set to _id, set the type parameter to ObjectId. If you set the type parameter to String, t
he data cannot be overwritten.
          },
          {
            "name": "age",
            "type": "int"
          },
          {
            "name": "id",
            "type": "long"
          },
          {
            "name": "wealth",
            "type": "double"
```

```
            },
            {
              "name": "hobby",
              "type": "array",
              "splitter": " "
            },
            {
              "name": "valid",
              "type": "boolean"
            },
            {
              "name": "date_of_join",
              "format": "yyyy-MM-dd HH:mm:ss",
              "type": "date"
            }
          ],
          "writeMode": {// The write mode.
            "isReplace": "true",
            "replaceKey": "_id"
          },
          "collectionName": "datax_test"// The name of the MongoDB collection.
        },
        "name": "Writer",
        "category": "writer"
      }
    ],
    "setting": {
      "errorLimit": {// The maximum number of dirty data records allowed.
        "record": "0"
      },
      "speed": {
        "jvmOption": "-Xms1024m -Xmx1024m",
        "throttle": true,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1,// The maximum number of concurrent threads.
        "mbps": "1"// The maximum transmission rate.
      }
    },
    "order": {
      "hops": [
```

```
      "hops": [
        {
          "from": "Reader",
          "to": "Writer"
        }
      ]
    }
}
```

2.4.5.4.12. Configure MySQL Writer

This topic describes the data types and parameters supported by MySQL Writer and how to configure it by using the codeless UI and code editor.

MySQL Writer allows you to write data to tables stored in MySQL databases. MySQL Writer connects to a remote MySQL database by using JDBC, and executes the `INSERT INTO` or `REPLACE INTO` statement to write data to the MySQL database. MySQL uses the InnoDB engine so that data is written to the database in batches.

> **Note**
> - You must configure a connection before you configure MySQL Writer.
> - MySQL Writer does not support MySQL 8.0 or later.

MySQL Writer can be used as a data migration tool by users such as DBAs. MySQL Writer obtains data from a Data Integration reader, and writes the data to the destination database based on value of the writeMode parameter.

> **Note** A sync node that uses MySQL Writer must have at least the permission to execute the `INSERT INTO or REPLACE INTO` statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

Data types

MySQL Writer supports most MySQL data types. Make sure that your data types are supported.

The following table lists the data types supported by MySQL Writer.

Category	MySQL data type
Integer	INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, and YEAR
Floating point	FLOAT, DOUBLE, and DECIMAL
String	VARCHAR, CHAR, TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT
Date and time	DATE, DATETIME, TIMESTAMP, and TIME

Category	MySQL data type
Boolean	BOOL
Binary	TINYBLOB, MEDIUMBLOB, BLOB, LONGBLOB, and VARBINARY

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
writeMode	The write mode. Valid values: • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. **Note** If you specify multiple SQL statements in the code editor, the system may not execute them in the same transaction.	No	None

Parameter	Description	Required	Default value
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor. Note If you specify multiple SQL statements in the code editor, the system may not execute them in the same transaction.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the MySQL database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure MySQL Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Solution to Primary Key Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.

GUI element	Description
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure MySQL Writer by using the code editor

In the following code, a node is configured to write data to a MySQL database. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"mysql",// The writer type
```

```
        "stepType" : "mysql",// The writer type.
        "parameter":{
          "postSql":[],// The SQL statement to execute after the sync node is run.
          "datasource":"",// The connection name.
          "column":[// The columns to which data is written.
            "id",
            "value"
          ],
          "writeMode":"insert",// The write mode.
          "batchSize":1024,// The number of data records to write at a time.
          "table":"",// The name of the destination table.
          "preSql":[]// The SQL statement to execute before the sync node is run.
        },
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{// The maximum number of dirty data records allowed.
        "record":"0"
      },
      "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling.
        "concurrent":1 // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.4.13. Configure Oracle Writer

This topic describes the data types and parameters supported by Oracle Writer and how to configure it by using the codeless UI and code editor.

Oracle Writer allows you to write data to tables stored in primary Oracle databases. Oracle Writer connects to a remote Oracle database by using JDBC, and executes an `INSERT INTO` statement to write data to the Oracle database.

> **Note** You must configure a connection before you configure Oracle Writer.

Oracle Writer is designed for ETL developers to import data from data warehouses to Oracle databases. Oracle Writer can also be used as a data migration tool by users such as DBAs.

Oracle Writer obtains data from a Data Integration reader, connects to a remote Oracle database by using JDBC, and then executes an SQL statement to write data to the Oracle database.

Data types

Oracle Writer supports most Oracle data types. Make sure that your data types are supported.

The following table lists the data types supported by Oracle Writer.

Category	Oracle data type
Integer	NUMBER, ROWID, INTEGER, INT, and SMALLINT
Floating point	NUMERIC, DECIMAL, FLOAT, DOUBLE PRECISION, and REAL
String	LONG, CHAR, NCHAR, VARCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, CHARACTER, CHARACTER VARYING, CHAR VARYING, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING, and NCHAR VARYING
Date and time	TIMESTAMP and DATE
Boolean	BIT and BOOLEAN
Binary	BLOB, BFILE, RAW, and LONG RAW

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None

Parameter	Description	Required	Default value
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. This means that all the field values of the original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Oracle database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Oracle Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Oracle Writer by using the code editor

In the following code, a node is configured to write data to an Oracle database.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"oracle",// The writer type.
      "parameter":{
        "postSql":[],// The SQL statement to execute after the sync node is run.
        "datasource":"",
        "session":[],// The settings of the session to the database.
        "column":[// The columns to which data is written.
          "id",
          "name"
        ],
        "encoding":"UTF-8",// The encoding format.
        "batchSize":1024,// The number of data records to write at a time.
        "table":"",// The name of the destination table.
        "preSql":[]// The SQL statement to execute before the sync node is run.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
```

```
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.4.5.4.14. Configure OSS Writer

This topic describes the data types and parameters supported by OSS Writer and how to configure it by using the codeless UI and code editor.

OSS Writer allows you to write one or more CSV-like files to OSS.

> **Note** You must configure a connection before you configure OSS Writer.

OSS Writer can write files that store logical two-dimensional tables, such as CSV files that store text data, to OSS.

OSS Writer allows you to convert data obtained from a Data Integration reader to files and write the files to OSS. The OSS files store unstructured data only. OSS Writer supports the following features:

- Writes only files that store text data. The text data must be logical two-dimensional tables.
- Writes CSV-like files with custom delimiters.
- Uses concurrent threads to write files. Each thread writes a file.
- Supports file rotation. OSS Writer can write data to another file when the size of the current file exceeds a specific value. OSS Writer can also write data to another file when the number of rows in the current file exceeds a specific value.

OSS Writer does not support the following features:

- Uses concurrent threads.to write a single file.
- Distinguishes between data types. OSS does not distinguish between data types. Therefore,

OSS Writer writes all data as strings to files in OSS.

The following table lists the data types supported by OSS Writer.

Category	OSS data type
Integer	LONG
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Date and time	DATE

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
object	The name prefix of the files to be written to OSS as objects. OSS simulates the directory effect by adding separators to object names. You can set the object parameter based on the following rules: • `"object": "datax"`: The names of the files start with datax, which is followed by a random string as the suffix. • `"object": "cdo/datax"`: The names of the files start with `/cdo/datax`, which is followed by a random string as the suffix. OSS uses backslashes (/) in objects to simulate the directory effect. If you do not want to add a random universally unique identifier (UUID) as the suffix, we recommend that you set the writeSingleObject parameter to `true`.	Yes	None

Parameter	Description	Required	Default value
writeMode	The mode in which OSS Writer writes the files. Valid values: • *truncate*: deletes all existing objects with the specified object name prefix before writing files to OSS. For example, if you set the object parameter to `abc`, all objects whose names start with abc are deleted. • *append*: writes all files and ensures that the actual file names do not conflict with those of existing objects by suffixing the file names with random UUIDs. For example, if you set the object parameter to DI, the actual names of the files written to OSS are in the following format: DI_****_****_****. • *nonConflict*: returns an error message if an object with the specified object name exists. For example, if you set the `object` parameter to abc and the object named abc123 exists, an error message is returned.	Yes	None
fileFormat	The format in which the files are written to OSS. Valid values: csv and text. • If a file is written as a CSV file, the file strictly follows CSV specifications. If the data in the file contains the column delimiter, the column delimiter is escaped by using double quotation marks (" "). • If a file is written as a text file, the data in the file is separated with the column delimiter. If the data in the file contains the column delimiter, the column delimiter is not escaped.	No	*text*
fieldDelimiter	The column delimiter that is used in the files to be written to OSS.	No	,
encoding	The encoding format of the files to be written to OSS.	No	*utf-8*
nullFormat	The string that represents null. No standard strings can represent null in text files. Therefore, Data Integration provides the nullFormat parameter to define which string represents a null pointer. For example, if you specify `nullFormat="null"`, Data Integration considers null as a null pointer.	No	None
header (advanced parameter, which cannot be set on the codeless UI)	The table header in the files to be written to OSS, for example, ['id','name','age'].	No	None

Parameter	Description	Required	Default value
maxFileSize (advanced parameter, which cannot be set on the codeless UI)	The maximum size of a single file that can be written to OSS. Default value: 100000. Unit: MB. File rotation based on this maximum size is similar to log rotation of Log4j. When a file is uploaded to OSS in multiple parts, the minimum size of a part is 10 MB. This size is the minimum granularity for file rotation. That is, if you set the maxFileSize parameter to less than 10 MB, the minimum size of a file is still 10 MB. Each call of the InitiateMultipartUploadRequest operation supports writing up to 10,000 parts. If file rotation occurs, suffixes, such as _1, _2, and _3, are appended to the new file names that consist of file name prefixes and random UUIDs.	No	100,000MB
suffix (advanced parameter, which cannot be set on the codeless UI)	The file name extension of the files to be written to OSS. For example, if you set the suffix parameter to .csv, the final name of a file written to OSS is in the format fileName****.csv.	No	None

Configure OSS Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Object Name Prefix	The object parameter in the preceding parameter description. Enter the path of the directory for storing the files. Do not include the name of the OSS bucket in the path.
File Type	The fileFormat parameter in the preceding parameter description. Valid values: *csv* and *text*.
Field Delimiter	The fieldDelimiter parameter in the preceding parameter description. The default delimiter is comma (,).
Encoding	The encoding parameter in the preceding parameter description. Default value: *UTF-8*.

Parameter	Description
Null String	The nullFormat parameter in the preceding parameter description. Enter a string that represents null. If the data in the source data store contains the string, the string is replaced with null.
Time Format	The format in which the data of the DATE type is serialized in an object, for example, `"dateFormat": "yyyy-MM-dd"`.
Solution to Duplicate Prefixes	The solution to take when a prefix conflict occurs. If an object with the specified name prefix exists, replace the object with the new object, insert the new object, or return an error message.

2. Configure field mapping. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure OSS Writer by using the code editor

In the following code, a node is configured to write files to OSS. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"oss",// The writer type.
      "parameter":{
        "nullFormat":"",// The string that represents null.
        "dateFormat":"",// The format in which the data of the DATE type is serialized in an object.
        "datasource":"",// The connection name.
        "writeMode":"",// The write mode.
        "encoding":"",// The encoding format.
        "fieldDelimiter":","// The column delimiter.
        "fileFormat":"",// The format in which the files are written to OSS.
        "object":""// The name prefix of the files to be written to OSS as objects.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
```

```
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1 // The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.4.5.4.15. Configure PostgreSQL Writer

This topic describes the data types and parameters supported by PostgreSQL Writer and how to configure it by using the codeless UI and code editor.

PostgreSQL Writer allows you to write data to a PostgreSQL database. PostgreSQL Writer connects to a remote PostgreSQL database by using JDBC, and executes an SQL statement to write data to the PostgreSQL database.

> **Note** You must configure a connection before you configure PostgreSQL Writer.

- PostgreSQL Writer generates the SQL statement based on the table, column, and where parameters that you specified, and sends the generated SQL statement to the PostgreSQL database.
- If you specify the querySql parameter, PostgreSQL Writer directly sends the value of this parameter to the PostgreSQL database.

Data types

PostgreSQL Writer supports most PostgreSQL data types. Make sure that your data types are supported.

The following table lists the data types supported by PostgreSQL Writer.

Data Integration data type	PostgreSQL data type
LONG	BIGINT, BIGSERIAL, INTEGER, SMALLINT, and SERIAL
DOUBLE	DOUBLE, PRECISION, MONEY, NUMERIC, and REAL

Data Integration data type	PostgreSQL data type
STRING	VARCHAR, CHAR, TEXT, BIT, and INET
DATE	DATE, TIME, and TIMESTAMP
BOOLEAN	BOOL
BYTES	BYTEA

Note

- Data types that are not listed in the table are not supported.
- You can convert the MONEY, INET, and BIT types by using syntax such as `a_inet::varchar`.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
writeMode	The write mode. Valid values: insert and copy. • *insert*: executes the `INSERT INTO` statement to write data to the PostgreSQL database. If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. We recommend that you select the insert mode. • *copy*: copies data between tables and the standard input or output file. Data Integration supports the `COPY FROM` command, which allows you to copy data from files to tables. We recommend that you try this mode when performance issues occur.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with commas (,), for example, `"column":["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None

Parameter	Description	Required	Default value
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the PostgreSQL database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Parameter	Description	Required	Default value
pgType	The PostgreSQL configuration for converting data types. Valid values: bigint[], double[], text[], jsonb, and json. Example: { "job": { "content": [{ "reader": {...}, "writer": { "parameter": { "column": [// The columns in the destination table to which data is written. "bigint_arr", "double_arr", "text_arr", "jsonb_obj", "json_obj"], "pgType": { // The PostgreSQL configuration for converting data types. In each key-value pair, the key specifies the name of a field in the destination table, and the value specifies the data type of the field. "bigint_arr": "bigint[]", "double_arr": "double[]", "text_arr": "text[]", "jsonb_obj": "jsonb", "json_obj": "json" }}] } } }] } }	No	None

Configure PostgreSQL Writer by using the codeless UI

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Write Method	The writeMode parameter in the preceding parameter description. Valid values: *Insert* and *Copy*.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

Button	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.

Parameter	Description
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure PostgreSQL Writer by using the code editor

In the following code, a node is configured to write data to a PostgreSQL database. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"postgresql",// The writer type.
      "parameter":{
        "postSql":[],// The SQL statement to execute after the sync node is run.
        "datasource":"// The connection name.
          "col1",
          "col2"
        ],
        "table":"",// The name of the destination table.
        "preSql":[]// The SQL statement to execute before the sync node is run.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
```

```
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1 // The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4.16. Configure Redis Writer

Redis Writer is a writer that is developed based on the Data Integration framework. It can be used to import data from data stores such as data warehouses to Redis databases.

Redis is a network-enabled key-value storage system that is either in-memory or permanent. It supports logs and delivers high performance. It can be used as a database, cache, and message broker. Redis supports diverse data types for values, including STRING, LIST, SET, ZSET (sorted set), and HASH.

Redis Writer interacts with a Redis server by using Jedis. As a preferred Java client development kit provided by Redis, Jedis supports almost all the features of Redis.

> **Note**
>
> - You must configure a connection before you configure Redis Writer.
> - If you write values of the LIST type to Redis by using Redis Writer, the result of rerunning a sync node is not idempotent. If the data type of the values is LIST, you must manually clear the corresponding data on Redis when you rerun a sync node.

Parameters

Parameter	Description	Required	Default value

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
keyIndexes	The columns used as the key. The index of the first column is 0. For example, if you want to set the first and second columns of the source data as the key, set the keyIndexes parameter to [0,1]. Note After you specify the keyIndexes parameter, Redis Writer specifies the remaining columns as the value. If you do not want to synchronize all the columns, filter columns when you configure the reader.	Yes	None
keyFieldDelimiter	The delimiter used to separate keys when data is written to Redis. Example: key=key1\u0001id. If multiple keys need to be concatenated, this parameter is required. If only one key exists, this parameter is not required.	No	*\u0001*
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Redis database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,000*
expireTime	The expiration time of the values to be cached in Redis. Unit: seconds. The data is valid permanently if you do not specify this parameter. • *seconds*: the relative time in seconds starting from the current time point. It specifies the time range during which data is valid. • *unixtime*: the UNIX timestamp, indicating that data is invalid at a specific time point in the future. The UNIX timestamp represents the number of seconds that have elapsed since 00:00:00 on January 1, 1970. Note If the specified expiration time is larger than 30 days, the server identifies the time as the UNIX timestamp.	No	*0*, indicating that the values never expire
timeout	The timeout period to connect to Redis when data is written to Redis. Unit: milliseconds.	No	*30,000*
dateFormat	The format in which the data of the DATE type is written to Redis. Set the value to yyyy-MM-dd HH:mm:ss.	No	None

Parameter	Description	Required	Default value
writeMode	The write mode. Redis supports diverse data types for values, including STRING, LIST, SET, ZSET (sorted set), and HASH. Redis Writer allows you to write values of the preceding types to Redis. The value of the writeMode parameter varies based on the specified data type of the values. Note When you configure Redis Writer, you can choose only one of the five data types described in the following table. If you do not specify a data type, the data type is STRING by default.	No	*string*

The following table lists the data types supported by Redis Writer.

Type	Parameter	Description	Required
String "writeMode":{ "type": "string", "mode": "set", "valueFieldDelimiter": "\u0001" }	type	The data type of the values is STRING.	Yes
	mode	The mode in which data of the STRING type is written to Redis.	Yes. Valid value: set (overwrites the existing data).
	valueFieldDelimiter	This parameter is required if two or more columns are specified as the values. This parameter is not required if only one column is specified as the values. The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: \u0001.

Type	Parameter	Description	Required
LIST "writeMode":{ "type": "list", "mode": "lpush|rpush", "valueFieldDelimiter": "\u0001" }	type	The data type of the values is LIST.	Yes
	mode	The mode in which data of the LIST type is written to Redis.	Yes. Valid values: lpush (stores the data at the leftmost of the list) and rpush (stores the data at the rightmost of the list).
	valueFieldDelimiter	The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: *\u0001*.
SET "writeMode":{ "type": "set", "mode": "sadd", "valueFieldDelimiter": "\u0001" }	type	The data type of the values is SET.	Yes
	mode	The mode in which data of the SET type is written to Redis.	Yes. Valid value: sadd (stores the data to a set, or overwrites the existing data).
	valueFieldDelimiter	The delimiter used to separate values if the data is of the STRING type. Example: value1\u0001value2\u0001value3.	No. Default value: *\u0001*.

Type	Parameter	Description	Required
ZSET (sorted set) "writeMode":{ "type": "zset", "mode": "zadd " }	type	The data type of the values is ZSET. Note If the data type of the values is ZSET, each data record must follow the following standard: Except for the key, a data record can contain only one score and one value. The score must be placed before the value. In this way, Redis Writer can identify which column is the score and which column is the value.	Yes
	mode	The mode in which data of the ZSET type is written to Redis.	Yes. Valid value: zadd (stores data to a sorted set, or overwrites the existing data).

Type	Parameter	Description	Required
HASH "writeMode":{ "type": "hash", "mode": "hset" }	type	The data type of the values is HASH. Note If the data type of the values is HASH, each data record must follow the following standards: Except for the key, a data record can contain only one attribute and one value. The attribute must be placed before the value. In this way, Redis Writer can identify which column is the attribute and which column is the value.	Yes
	mode	The mode in which data of the HASH type is written to Redis.	Yes. Valid value: hmset (stores data to a hash sorted set, or overwrites the existing data). If you do not specify a data type, the data type is STRING by default.

Configure Redis Writer by using the codeless UI

The codeless UI is not supported for Redis Reader.

Configure Redis Writer by using the code editor

In the following code, a node is configured to write data to Redis. For more information about parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
```

```
        "parameter":{},
        "name":"Reader",
        "category":"reader"
      },
      {
        "stepType":"redis",// The writer type.
        "parameter":{
          "expireTime":{// The expiration time of the values to be cached in Redis.
            "seconds":"1000"
          },
          "keyFieldDelimiter":"u0001",// The delimiter used to separate keys when data is written to Redis.
          "dateFormat":"yyyy-MM-dd HH:mm:ss",// The format in which the data of the DATE type is written to Redis.
          "datasource":"",// The connection name.
          "writeMode":{// The write mode.
            "mode":"",// The write mode used to write data of a specified data type.
            "valueFieldDelimiter":"",// The delimiter used to separate values.
            "type":""// The data type of the values.
          },
          "keyIndexes":[// The columns used as the key.
            0,
            1
          ],
          "batchSize":"1000"// The number of data records to write at a time.
        },
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1 // The maximum number of concurrent threads.
      }
    },
```

```
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}Writer"
      }
    ]
  }
}
```

2.4.5.4.17. Configure SQL Server Writer

This topic describes the data types and parameters supported by SQL Server Writer and how to configure it by using the codeless UI and code editor.

SQL Server Writer allows you to write data to tables stored in primary SQL Server databases. SQL Server Writer connects to a remote SQL Server database by using JDBC, and executes an `INSERT INTO` statement to write data to the SQL Server database. Internally, data is submitted to the database in batches.

SQL Server Writer is designed for ETL developers to import data from data warehouses to SQL Server databases. SQL Server Writer can also be used as a data migration tool by users such as DBAs.

SQL Server Writer obtains data from a Data Integration reader, and generates the `INSERT INTO` statement. If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows. To improve performance, SQL Server Writer makes batch updates with the `PreparedStatement` method and sets `rewriteBatchedStatements` to true. In this way, SQL Server Writer buffers data, and submits a write request when the amount of data in the buffer reaches a specific threshold.

> **Note**
>
> - Data can be written only to tables stored in the primary SQL Server database.
> - A sync node that uses SQL Server Writer must have at least the permission to execute the INSERT INTO statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

Data types

SQL Server Writer supports most SQL Server data types. Make sure that your data types are supported.

The following table lists the data types supported by SQL Server Writer.

Category	SQL Server data type
Integer	BIGINT, INT, SMALLINT, and TINYINT
Floating point	FLOAT, DECIMAL, REAL, and NUMERIC
String	CHAR, NCHAR, NTEXT, NVARCHAR, TEXT, VARCHAR, NVARCHAR (MAX), and VARCHAR (MAX)
Date and time	DATE, TIME, and DATETIME
Boolean	BIT
Binary	BINARY, VARBINARY, VARBINARY (MAX), and TIMESTAMP

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the destination table.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with commas (,). Example: `"column": ["id","name","age"]`. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`.	Yes	None
preSql	The SQL statement to execute before the sync node is run. For example, you can clear outdated data before data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
postSql	The SQL statement to execute after the sync node is run. For example, you can add a timestamp after data synchronization. You can execute only one SQL statement on the codeless UI and multiple SQL statements in the code editor.	No	None
writeMode	The write mode. Valid value: *insert*. When a data record violates the primary key constraint or unique index constraint, Data Integration considers it dirty and retains the original data.	No	*insert*

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the SQL Server database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure SQL Serve Writer by using the codeless UI

1. Configure the connections.

   Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Statement Run Before Writing	The preSql parameter in the preceding parameter description. Enter an SQL statement to execute before the sync node is run.
Statement Run After Writing	The postSql parameter in the preceding parameter description. Enter an SQL statement to execute after the sync node is run.
Solution to Primary Key Violation	The writeMode parameter in the preceding parameter description. Select the required write mode.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description.

   Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field, or move the pointer over a field and click the **Delete** icon to delete the field.

GUI element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

GUI element	Description
Change Fields	Click the Change Fields icon. In the Change Fields dialog box, you can manually edit fields in the source table. Each field occupies a row. The first and the last blank rows are included, whereas other blank rows are ignored.
Add	○ Click Add to add a field. You can enter constants. Each constant must be enclosed in single quotation marks (' '), for example, 'abc' and '123'. ○ You can use scheduling parameters such as ${bizdate}. ○ You can enter functions supported by relational databases, for example, now() and count(1). ○ If the value you entered cannot be parsed, the type is displayed as Unidentified.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure SQL Server Writer by using the code editor

In the following code, a node is configured to write data to an SQL Server database. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{}
```

```
        "parameter":{},
        "name":"Reader",
        "category":"reader"
      },
      {
        "stepType":"sqlserver",// The writer type.
        "parameter":{
          "postSql":[],// The SQL statement to execute after the sync node is run.
          "datasource":"",// The connection name.
          "column":[// The columns to which data is written.
            "id",
            "name"
          ],
          "table":"",// The name of the destination table.
          "preSql":[]// The SQL statement to execute before the sync node is run.
        },
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1 // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.4.18. Configure Elasticsearch Writer

This topic describes the data types and parameters supported by Elasticsearch Writer and how to configure it by using the code editor.

Elasticsearch is an open-source product that complies with the Apache open standards. It is the mainstream search engine for enterprise data. Elasticsearch is a Lucene-based data search and analysis tool that provides distributed services. The mappings between Elasticsearch core concepts and database core concepts are as follows:

```
Relational database (instance) -> database -> table -> row -> column
Elasticsearch -> index -> type -> document -> field
```

Elasticsearch can contain multiple indexes (databases). Each index can contain multiple types (tables). Each type can contain multiple documents (rows). Each document can contain multiple fields (columns). Elasticsearch Writer uses the RESTful API of Elasticsearch to write multiple data records retrieved by a reader to Elasticsearch at a time.

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint for accessing Elasticsearch, in the format of `http://xxxx.com:9999`.	No	None
accessId	The AccessKey ID for accessing Elasticsearch, which is used for authorization when a connection with Elasticsearch is established. **Note** The accessId and accessKey parameters are required. If you do not set the parameters, an error is returned. If you use on-premises Elasticsearch for which basic authentication is not configured, the AccessKey ID and AccessKey secret are not required. In this case, you can set the accessId and accessKey parameters to random values.	No	None
accessKey	The AccessKey secret for accessing Elasticsearch.	No	None
index	The index name in Elasticsearch.	No	None
indexType	The type name in the index of Elasticsearch.	No	*Elasticsearch*
cleanup	Specifies whether to clear existing data in the index. The method used to clear the data is to delete and rebuild the corresponding index. The default value false indicates that the existing data in the index is retained.	No	*false*

Parameter	Description	Required	Default value
batchSize	The number of data records to write at a time.	No	*1000*
trySize	The number of retries after a failure.	No	*30*
timeout	The connection timeout of the client. Unit: milliseconds.	No	*600000*
discovery	Specifies whether to enable Node Discovery. When Node Discovery is enabled, the server list in the client is polled and regularly updated.	No	*false*
compression	Specifies whether to enable compression for an HTTP request.	No	*true*
multiThread	Specifies whether to use multiple threads for an HTTP request.	No	*true*
ignoreWriteError	Specifies whether to ignore write errors and proceed with writing without retries.	No	*false*
ignoreParseError	Specifies whether to ignore format parsing errors and proceed with writing.	No	*true*
alias	The alias of the index. The alias feature of Elasticsearch is similar to the view feature of a traditional database. For example, if you create an alias named my_index_alias for the index my_index, the operations on my_index_alias also take effect on my_index. Configuring alias means that after the data import is completed, an alias is created for the specified index.	No	None
aliasMode	The mode in which an alias is added after the data is imported. Valid values: *append* and *exclusive*.	No	append

Parameter	Description	Required	Default value
settings	The delimiter (-,-) for splitting the source data if you are inserting an array to Elasticsearch. Example: The source column stores data `a-,-b-,-c-,-d` of the String type. Elasticsearch Writer uses the delimiter (-,-) to split the source data and obtains the array `["a", "b", "c", "d"]`. Then, Elasticsearch Writer writes the array to the corresponding field in Elasticsearch.	No	-,-
	The fields of the document. The parameters for each field include basic parameters such as name and type and advanced parameters such as analyzer, format, and array. The field types supported by Elasticsearch are as follows:		

Parameter	Description	Required	Default value
column	- id // The id type corresponds to the _id type in Elasticsearch, and can be considered as the unique primary key. Data with the same ID will be overwritten and not indexed. - string - text - keyword - long - integer - short - byte - double - float - date - boolean - binary - integer_range - float_range - long_range - double_range - date_range - geo_point - geo_shape - ip - token_count - array - object - nested • When the field type is Text, you can specify the analyzer, norms, and index_options parameters. Example: `{ "name": "col_text", "type": "text", "analyzer": "ik_max_word" }` • When the field type is date, you can specify the format and timezone parameters, indicating the date serialization format and the time zone, respectively. Example:	Yes	None

Parameter	Description	Required	Default value
	{ "name": "col_date", "type": "date", "format": "yyyy-MM-dd HH:mm:ss", "timezone": "UTC" } • When the field type is ge_shape, you can specify the tree (geohash or quadtree) and precision parameters. Example: { "name": "col_geo_shape", "type": "geo_shape", "tree": "quadtree", "precision": "10m" }		
dynamic	Specifies whether to use the mapping configuration of Elasticsearch. A value of *true* indicates that the mapping configuration of Elasticsearch, instead of the mapping configuration of Data Integration, is used.	No	*false*
	If you specify the array parameter for a field and set the array parameter to *true*, the field is an array column. Elasticsearch Writer uses the delimiter specified by the splitter to split the source data, converts the data to an array of strings, and writes the array to the destination. Only one delimiter is supported for one node. Example: { "name": "col_integer_array", "type": "integer", "array": true }		

Parameter	Description	Required	Default value
actionType	The type of the action for writing data to Elasticsearch. Currently, Data Integration supports only the following action types: *index* and *update*. Default value: *index*. • *index*: Data Integration uses Index.Builder of the Elasticsearch SDK to construct a request for writing multiple data records at a time. In *index* mode, Elasticsearch first checks whether an ID is specified for the document to be inserted. ◦ If the ID is not specified, Elasticsearch generates a unique ID by default. In this case, the document is directly inserted to Elasticsearch. ◦ If the ID is specified, Elasticsearch replaces the existing document with the document to be inserted. Note In this case, you cannot modify specific fields in the document. • *update*: Data Integration uses Update.Builder of the Elasticsearch SDK to construct a request for writing multiple data records at a time. In *update* mode, Elasticsearch calls the get method of InternalEngine to obtain the information of the original document for each update. In this way, you can modify specific fields. In update mode, you must obtain the information of the original document for each update, which greatly affects the performance. However, you can modify specific fields in this mode. If the original document does not exist, the new document is directly inserted.	No	*index*

Configure Elasticsearch Writer by using the code editor

In the following code, a node is configured to write data to Elasticsearch. For more information about the parameters, see the preceding parameter description.

```
{
  "order": {
    "hops": [
      {
        "from": "Reader",
        "to": "Writer"
      }
```

```
    ]
  },
  "setting": {
    "errorLimit": {
      "record": "0"
    },
    "speed": {
      "concurrent": 1,
      "throttle": false
    }
  },
  "steps": [
    {
      "category": "reader",
      "name": "Reader",
      "parameter": {

      },
      "stepType": "stream"
    },
    {
      "category": "writer",
      "name": "Writer",
      "parameter": {
        "endpoint": "http://xxxx.com:9999",
        "accessId": "xxxx",
        "accessKey": "yyyy",
        "index": "test-1",
        "type": "default",
        "cleanup": true,
        "settings": {
          "index": {
            "number_of_shards": 1,
            "number_of_replicas": 0
          }
        },
        "discovery": false,
        "batchSize": 1000,
        "splitter": ",",
        "column": [
          {
```

```
{
  "name": "pk",
  "type": "id"
},
{
  "name": "col_ip",
  "type": "ip"
},
{
  "name": "col_double",
  "type": "double"
},
{
  "name": "col_long",
  "type": "long"
},
{
  "name": "col_integer",
  "type": "integer"
},
{
  "name": "col_keyword",
  "type": "keyword"
},
{
  "name": "col_text",
  "type": "text",
  "analyzer": "ik_max_word"
},
{
  "name": "col_geo_point",
  "type": "geo_point"
},
{
  "name": "col_date",
  "type": "date",
  "format": "yyyy-MM-dd HH:mm:ss"
},
{
  "name": "col_nested1",
  "type": "nested"
```

```
            },
            {
              "name": "col_nested2",
              "type": "nested"
            },
            {
              "name": "col_object1",
              "type": "object"
            },
            {
              "name": "col_object2",
              "type": "object"
            },
            {
              "name": "col_integer_array",
              "type": "integer",
              "array": true
            },
            {
              "name": "col_geo_shape",
              "type": "geo_shape",
              "tree": "quadtree",
              "precision": "10m"
            }
          ]
        },
        "stepType": "elasticsearch"
      }
    ],
    "type": "job",
    "version": "2.0"
}
```

> **Note** Currently, Elasticsearch that is deployed in a Virtual Private Cloud (VPC) supports only custom resource groups. A sync node that is run on the default resource group may fail to connect to Elasticsearch.

2.4.5.4.19. Configure LogHub Writer

This topic describes the data types and parameters supported by LogHub Writer and how to configure it by using the code editor.

LogHub Writer allows you to transfer data from a Data Integration reader to LogHub through Log Service Java SDK.

> **Note** LogHub does not guarantee idempotence. Rerunning a node after the node fails may result in redundant data.

LogHub Writer obtains data from a Data Integration reader and converts the data types supported by Data Integration to String. When the number of the data records reaches the value specified for the batchSize parameter, LogHub Writer sends the data records to LogHub at a time through Log Service Java SDK. LogHub Writer sends 1,024 data records at a time by default. The batchSize parameter can be set to 4096 at most.

Data types

The following table lists the data types supported by LogHub Writer.

Data Integration data type	LogHub data type
LONG	STRING
DOUBLE	STRING
STRING	STRING
DATE	STRING
BOOLEAN	STRING
BYTES	STRING

Parameters

Parameter	Description	Required	Default value
endpoint	The endpoint for accessing Log Service.	Yes	None
accessKeyId	The AccessKey ID for accessing Log Service.	Yes	None
accessKeySecret	The AccessKey secret for accessing Log Service.	Yes	None
project	The name of the destination Log Service project.	Yes	None
logstore	The name of the destination Logstore.	Yes	None
topic	The name of the destination topic.	No	Empty string
batchSize	The number of data records to write at a time.	No	*1024*
column	The columns in each data record.	Yes	None

Configure LogHub Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for LogHub Writer.

Configure LogHub Writer by using the code editor

In the following code, a node is configured to write data to LogHub. For more information about the parameters, see the preceding parameter description.

```
{
  "type": "job",
  "version": "2.0",// The version number.
  "steps": [
    { //
      "stepType": "stream",
      "parameter": {},
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "loghub",// The writer type.
      "parameter": {
        "datasource": "",// The connection name.
        "column": [// The columns in each data record.
          "col0",
          "col1",
          "col2",
          "col3",
          "col4",
          "col5"
        ],
        "topic": "",// The name of the destination topic.
        "batchSize": "1024",// The number of data records to write at a time.
        "logstore": ""// The name of the destination Logstore.
      },
      "name": "Writer",
      "category": "writer"
    }
  ],
  "setting": {
    "errorLimit": {
      "record": ""// The maximum number of dirty data records allowed.
    },
```

```
        "speed": {
            "concurrent": 3,// The maximum number of concurrent threads.
            "throttle": false,// Specifies whether to enable bandwidth throttling. A value of false indicates that the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maximum transmission rate takes effect only if you set this parameter to true.
        }
    },
    "order": {
        "hops": [
            {
                "from": "Reader",
                "to": "Writer"
            }
        ]
    }
}
```

2.4.5.4.20. Configure Open Search Writer

This topic describes the data types and parameters supported by Open Search Writer and how to configure it by using the code editor.

How it works

Open Search Writer allows you to insert data to or update data in Open Search. Open Search Writer is designed for developers to import data to Open Search so that the data can be searched.

Specifically, Open Search Writer uses the search API provided by Open Search to import data.

> **Note**
>
> - Open Search V3 uses internal dependent databases, with POM of `com.aliyun.opensearch aliyun-sdk-opensearch 2.1.3`.
> - To use Open Search Writer, you must install JDK 1.6-32 or later. You can run the `java-version` command to view the JDK version.

Features

The columns in Open Search are unordered. Open Search Writer writes data strictly in accordance with the order of the specified columns. If the number of specified columns is less than that in Open Search, redundant columns in Open Search are set to the default value or null.

Assume that an Open Search table contains columns a, b, and c, and you only need to write data to columns b and c. You can set the column parameter to ["c","b"]. In this case, Open Search Writer imports the first and second columns of the source data obtained from a reader to columns c and b in the Open Search table respectively. Column a in the Open Search table is set to the default value or null.

Additional instructions:

- Handling of column configuration errors

  To avoid losing the data of redundant columns and ensure high data reliability, Open Search Writer returns an error message if the number of columns to be written is more than that in the destination Open Search table. For example, if an Open Search table contains columns a, b, and c, Open Search Writer returns an error if more than three columns are to be written to the table.

- Table configuration

  Open Search Writer can write data to only one table at a time.

- Node rerunning

  After a node is rerun, data is automatically overwritten based on IDs. Therefore, the data written to Open Search must contain one ID column. An ID is a unique identifier of a row in Open Search. The existing data with the same ID as the new data will be overwritten.

- Node rerunning

  After a node is rerun, data is automatically overwritten based on IDs.

Data types

Open Search Writer supports most Open Search data types. Make sure that your data types are supported.

The following table lists the data types supported by Open Search Writer.

Category	Open Search data type
Integer	INT
Floating point	DOUBLE and FLOAT
String	TEXT, LITERAL, and SHORT_TEXT
Date and time	INT
Boolean	LITERAL

Parameters

Parameter	Description	Required	Default value
accessId	The AccessKey ID for connecting to the Open Search database.	Yes	None

Parameter	Description	Required	Default value
accessKey	The AccessKey secret for connecting to the Open Search database.	Yes	None
host	The endpoint for connecting to Open Search. You can view the endpoint in the Apsara Stack console.	Yes	None
indexName	The name of the Open Search project.	Yes	None
table	The name of the table to which data is written. You can specify only one table name because Data Integration does not support importing data to multiple tables at a time.	Yes	None
column	The columns in the destination table to which data is written. To write data to all the columns in the destination table, set the value to an asterisk (*), for example, `"column":["*"]`. Separate the columns with a comma (,) if data is written to some of the columns in the destination table. Example: `"column":["id","name"]`. Open Search Writer supports filtering columns and changing the order of columns. Assume that an Open Search table has three columns: a, b, and c. If you want to write data only to columns c and b, you can set the column parameter in the format `"column":["c","b"]`. During data synchronization, column a is automatically set to null.	Yes	None
batchSize	The number of data records to write at a time. Data is written to Open Search in batches. The advantage of Open Search is data query. The transactions per second (TPS) of Open Search is generally not high. Set this parameter based on the resources available for the account that is used to connect to Open Search. Generally, the size of a data record must be less than 1 MB, and the size of the data records to write at a time must be less than 2 MB.	Required only for writing data to a partitioned table	*300*

Parameter	Description	Required	Default value
writeMode	The write mode. To ensure the idempotence of write operations, set the writeMode parameter to add/update when you configure Open Search Writer. • add: deletes the existing data record and inserts the new data record to Open Search, which is an atomic operation. • update: updates the existing data record based on the new data record, which is an atomic operation. Note Writing data to Open Search in batches is not an atomic operation. Part of the data may fail to be written. Exercise caution when you set the writeMode parameter. Open Search V3 does not support the update mode.	Yes	None
ignoreWriteError	Specifies whether to ignore failed write operations. Example: `"ignoreWriteError":true`. If data is written to Open Search in batches, this parameter specifies whether to ignore failed write operations in the current batch. If you set the parameter to true, Open Search Writer continues to perform other write operations. If you set the parameter to false, the sync node ends and an error message is returned. We recommend that you use the default value.	No	*false*
version	The version of Open Search, for example, `"version":"v3"`. We recommend that you use Open Search V3 because the push operation faces many constraints in Open Search V2.	No	*v2*

Configure Open Search Writer by using the code editor

In the following code, a node is configured to write data to Open Search.

```
{
  "type": "job",
  "version": "1.0",
  "configuration": {
    "reader": {},
    "writer": {
      "plugin": "opensearch",
      "parameter": {
        "accessId": "*********",
        "accessKey": "********",
        "host": "http://yyyy.aliyuncs.com",
        "indexName": "datax_xxx",
        "table": "datax_yyy",
        "column": [
        "appkey",
        "id",
        "title",
        "gmt_create",
        "pic_default"
        ],
        "batchSize": 500,
        "writeMode": add,
        "version":"v2",
        "ignoreWriteError": false
      }
    }
  }
}
```

2.4.5.4.21. Configure Table Store Writer

This topic describes the data types and parameters supported by Table Store Writer and how to configure it by using the code editor.

Table Store is a NoSQL database service built on the Apsara distributed operating system that allows you to store and access large amounts of structured data in real time. Table Store organizes data into instances and tables. Using data sharding and load balancing technologies, Table Store seamlessly expands the data scale.

Table Store Writer connects to a Table Store server by using the official Java SDK and writes data to the Table Store server by using the SDK. Table Store Writer has greatly optimized the write process, including retry upon write timeout, retry upon exceptions, and batch submission.

Currently, Table Store Writer supports all Table Store data types and supports the following two write modes:

- PutRow: the PutRow API operation for Table Store, which is used to insert data to a specified row. If this row does not exist, a new row is added. Otherwise, the original row is overwritten.
- UpdateRow: the UpdateRow API operation for Table Store, which is used to update the data of a specified row. If this row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or deleted as requested.

Currently, Table Store Writer supports all Table Store data types. The following table lists the data types supported by Table Store Writer.

Category	Table Store data type
Integer	INTEGER
Floating point	DOUBLE
String	STRING
Boolean	BOOLEAN
Binary	BINARY

Note To write data of the INTEGER type, set the data type to INT in the code editor. Then, DataWorks converts the INT type to the Integer type. If you set the data type to INTEGER for the data to be written to Table Store, an error is reported in the log and the sync node fails.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
endPoint	The endpoint of the Table Store server.	Yes	None
accessId	The AccessKey ID for accessing Table Store.	Yes	None
accessKey	The AccessKey secret for accessing Table Store.	Yes	None
instanceName	The name of the Table Store instance to access. You access and manage the Table Store service through an instance. After activating Table Store, you can create an instance in the Table Store console and then create and manage tables in the instance. Instances are the basic unit for managing Table Store resources. All access control and resource measurement for applications are completed at the instance level.	Yes	None

Parameter	Description	Required	Default value
table	The name of the destination table. You can specify only one table as the destination table. Multi-table synchronization is not required for Table Store.	Yes	None
primaryKey	The primary keys of the destination table in Table Store. The primary keys are described in a JSON array. Table Store is a NoSQL database service. The field names must be specified for Table Store Writer to import data. Note The primary keys in Table Store only support the STRING and INT types. Therefore, you must set the data type of a primary key to either of the two types in the code editor. Data Integration supports converting data types. Table Store Writer can convert data that is not of the STRING or INT type to the STRING or INT type. Example: "primaryKey" : [{"name":"pk1", "type":"string"}, {"name":"pk2", "type":"int"}],	Yes	None
column	The columns in the destination table to which data is written. The columns are described in a JSON array. Format: {"name":"col2", "type":"INT"}, The name parameter specifies the name of the column to which data is written. The type parameter specifies the data type of the column. Data types supported by Table Store include String, Int, Double, Boolean, and Binary.	Yes	None

Parameter	Description	Required	Default value
writeMode	The write mode. Constants, functions, or custom statements are not supported during the write process. The following three modes are supported: • Single-row operations ○ GetRow: reads data from a single row. ○ PutRow: the PutRow API operation for Table Store, which is used to insert data to a specified row. If this row does not exist, a new row is added. Otherwise, the original row is overwritten. ○ UpdateRow: the UpdateRow API operation for Table Store, which is used to update the data of a specified row. If this row does not exist, a new row is added. Otherwise, the values of the specified columns are added, modified, or deleted as requested. ○ DeleteRow: deletes a row. • Multi-row operation BatchGetRow: reads data from multiple rows. • Range-based operation GetRange: reads data from a table within a range.	Yes	None

Configure Table Store Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Table Store Writer.

Configure Table Store Writer by using the code editor

In the following code, a node is configured to write data to Table Store.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"ots",// The writer type.
```

```
    "parameter":{
      "datasource":"",// The connection name.
      "column":[// The columns to which data is written.
        {
          "name":"columnName1",// The name of the column.
          "type": "INT" // The data type of the column.
        },
        {
          "name":"columnName2",
          "type":"STRING"
        },
        {
          "name":"columnName3",
          "type":"DOUBLE"
        },
        {
          "name":"columnName4",
          "type":"BOOLEAN"
        },
        {
          "name":"columnName5",
          "type":"BINARY"
        }
      ],
      "writeMode":"",// The write mode.
      "table":"",// The name of the destination table.
      "primaryKey":[// The primary keys of the destination table in Table Store.
        {
          "name":"pk1",
          "type":"STRING"
        },
        {
          "name":"pk2",
          "type":"INT"
        }
      ]
    },
    "name":"Writer",
    "category":"writer"
  }
```

```
    ],
    "setting":{
        "errorLimit":{
            "record":"0"// The maximum number of dirty data records allowed.
        },
        "speed":{
            "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
            "concurrent":1,// The maximum number of concurrent threads.
        }
    },
    "order":{
        "hops":[
            {
                "from":"Reader",
                "to":"Writer"
            }
        ]
    }
}
```

2.4.5.4.22. Configure RDBMS Writer

This topic describes the data types and parameters supported by RDBMS Writer and how to configure it by using the code editor.

RDBMS Writer allows you to write data to tables stored in primary relational database management system (RDBMS) databases. Specifically, RDBMS Writer obtains data from a Data Integration reader, connects to a remote RDBMS database through Java Database Connectivity (JDBC), and then runs an `INSERT INTO` statement to write data to the RDBMS database. RDBMS Writer is a common writer for relational databases. To enable RDBMS Writer to support a new relational database, register the driver for the relational database.

RDBMS Writer is designed for extract-transform-load (ETL) developers to import data from data warehouses to RDBMS databases. RDBMS Writer can also be used as a data migration tool by users such as database administrators (DBAs).

Data types

RDBMS Writer supports most data types in relational databases, such as numbers and characters. Make sure that your data types are supported.

Parameters

Parameter	Description	Required	Default value
jdbcUrl	The JDBC URL for connecting to the database. The format must be in accordance with official specifications. You can also specify the information of the attachment facility. The format varies with the database type. Data Integration selects an appropriate driver for data reading based on the format. • Format for DM databases: `jdbc:dm://ip:port/database` • Format for Db2 databases: `jdbc:db2://ip:port/database` • Format for PPAS databases: `jdbc:edb://ip:port/database`	Yes	None
username	The username for connecting to the database.	Yes	None
password	The password for connecting to the database.	Yes	None
table	The name of the destination table.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Note We recommend that you do not use the default setting.	Yes	None
preSql	The SQL statement to run before the sync node is run. For example, you can clear outdated data before data synchronization. Currently, you can run only one SQL statement. Note If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None

Parameter	Description	Required	Default value
postSql	The SQL statement to run after the sync node is run. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement. Note If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the RDBMS database over the network, and increase the throughput. However, an excessively large value may lead to the out of memory (OOM) error during the data synchronization process.	No	1024

Configure RDBMS Writer by using the code editor

In the following code, a node is configured to write data to an RDBMS database.

```
{
  "job": {
    "setting": {
      "speed": {
        "channel": 1
      }
    },
    "content": [
      {
        "reader": {
          "name": "streamreader",
          "parameter": {
            "column": [
              {
                "value": "DataX",
                "type": "string"
              },
              {
                "value": 19880808,
```

```
                "type": "long"
              },
              {
                "value": "1988-08-08 08:08:08",
                "type": "date"
              },
              {
                "value": true,
                "type": "bool"
              },
              {
                "value": "test",
                "type": "bytes"
              }
            ],
            "sliceRecordCount": 1000
          }
        },
        "writer": {
          "name": "RDBMS Writer",
          "parameter": {
            "connection": [
              {
                "jdbcUrl": "jdbc:dm://ip:port/database",
                "table": [
                  "table"
                ]
              }
            ],
            "username": "username",
            "password": "password",
            "table": "table",
            "column": [
              "*"
            ],
            "preSql": [
              "delete from XXX;"
            ]
          }
        }
      }
```

```
        }
      ]
    }
}
```

You can enable RDBMS Writer to support a new database as follows:

1. Go to the directory of RDBMS Writer, *${DATAX_HOME}/plugin/writer/RDBMS Writer*. In the preceding directory, ${DATAX_HOME} indicates the main directory of Data Integration.
2. Add the driver of your database to the drivers array in the *plugin.json* file in the RDBMS Writer directory. RDBMS Writer automatically selects an appropriate driver for connecting to a database.

   ```
   {
     "name": "RDBMS Writer",
     "class": "com.alibaba.datax.plugin.reader.RDBMS Writer.RDBMS Writer",
     "description": "useScene: prod. mechanism: Jdbc connection using the database, execute select sql, retrieve data from the ResultSet. warn: The more you know about the database, the less pro blems you encounter.",
     "developer": "alibaba",
     "drivers": [
       "dm.jdbc.driver.DmDriver",
       "com.ibm.db2.jcc.DB2Driver",
       "com.sybase.jdbc3.jdbc.SybDriver",
       "com.edb.Driver"
     ]
   }
   ```

3. Add the package of the driver to the libs directory in the RDBMS Writer directory.

```
$tree
.
|-- libs
|   |-- Dm7JdbcDriver16.jar
|   |-- commons-collections-3.0.jar
|   |-- commons-io-2.4.jar
|   |-- commons-lang3-3.3.2.jar
|   |-- commons-math3-3.1.1.jar
|   |-- datax-common-0.0.1-SNAPSHOT.jar
|   |-- datax-service-face-1.0.23-20160120.024328-1.jar
|   |-- db2jcc4.jar
|   |-- druid-1.0.15.jar
|   |-- edb-jdbc16.jar
|   |-- fastjson-1.1.46.sec01.jar
|   |-- guava-r05.jar
|   |-- hamcrest-core-1.3.jar
|   |-- jconn3-1.0.0-SNAPSHOT.jar
|   |-- logback-classic-1.0.13.jar
|   |-- logback-core-1.0.13.jar
|   |-- plugin-rdbms-util-0.0.1-SNAPSHOT.jar
|   `-- slf4j-api-1.7.10.jar
|-- plugin.json
|-- plugin_job_template.json
`-- RDBMS Writer-0.0.1-SNAPSHOT.jar
```

2.4.5.4.23. Configure Stream Writer

This topic describes the data types and parameters supported by Stream Writer and how to configure it by using the code editor.

Stream Writer allows you to display the data obtained from a Data Integration reader on the screen or discard the data. Stream Writer is mainly applicable to performance testing for data synchronization and basic functional testing.

Parameters

print

- Description: specifies whether to display the data obtained from the reader on the screen.
- Required: No
- Default value: true

Configure Stream Writer by using the codeless UI

Currently, the codeless user interface (UI) is not supported for Stream Writer.

Configure Stream Writer by using the code editor

In the following code, a node is configured to display the data obtained from a Data Integration reader on the screen.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
    {
      "stepType":"stream",// The writer type.
      "parameter":{
        "print": false, // Specifies whether to display data on the screen.
        "fieldDelimiter":","// The column delimiter.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{
      "record":"0"// The maximum number of dirty data records allowed.
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.4.24. Configure Hive Writer

Hive Writer allows you to write data to HDFS and load the data to Hive. This topic describes how Hive Writer works, its parameters, and how to configure it by using the codeless UI and code editor.

Background information

Hive is a Hadoop-based data warehouse tool that is used to process large amounts of structured logs. Hive maps structured data files to a table and allows you to execute SQL statements to query data in the table.

Essentially, Hive converts Hive Query Language (HQL) or SQL statements to MapReduce programs.

- Hive stores processed data in HDFS.
- Hive uses MapReduce programs to analyze data at the underlying layer.
- Hive runs MapReduce programs on Yarn.

How it works

Hive Writer accesses a Hive metastore, parses the configuration to obtain the file storage path, file format, and column delimiter of the file to which data is written, and then writes data to the HDFS file. Hive Writer loads data in the HDFS file to the destination Hive table by using JDBC.

The underlying logic of Hive Writer is the same as that of HDFS Writer. You can configure parameters of HDFS Writer in the parameters of Hive Writer. Data Integration transparently transmits the configured parameters to HDFS Writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection.	Yes	None
column	The columns to which data is written. Example: `"column": ["id","name"]`. • Column pruning is supported. You can select specific columns to export. • The column parameter must explicitly specify a set of columns to which data is written. The parameter cannot be left empty. • The column order cannot be changed.	Yes	None
table	The name of the Hive table to which data is written. Note The name is case-sensitive.	Yes	None

Parameter	Description	Required	Default value
partition	The partition in the Hive table to which data is written. • This parameter is required for a partitioned Hive table. The sync node writes data to the partition specified by the partition parameter. • This parameter is not required for a non-partitioned table.	No	None
writeMode	The mode in which data is loaded to the Hive table. After data is written to the HDFS file, Hive Writer executes the `LOAD DATA INPATH (overwrite) INTO TABLE` statement to load data to the Hive table. The writeMode parameter specifies the data loading mode. • *truncate*: deletes existing data before loading the data to the Hive table. • *append*: retains the existing data and appends the data to the Hive table. • If the writeMode parameter is set to Other, the data is written to the HDFS file but not loaded to the Hive table. Note Setting the writeMode parameter is a high-risk operation. Pay attention to the destination directory and the value of this parameter to avoid deleting data incorrectly. This parameter must be used together with the hiveConfig parameter.	Yes	None

Parameter	Description	Required	Default value
hiveConfig	The extended parameters for Hive, including hiveCommand, jdbcUrl, username, and password. • hiveCommand: the full path of the Hive client. After you run the `hive -e` command, the `LOAD DATA INPATH` statement is executed to load data based on the mode specified by the writeMode parameter. The client specified by the hiveCommand parameter provides access information about Hive. • jdbcUrl, username, and password: the information that is required to connect to Hive by using JDBC. After Hive Writer connects to Hive by using JDBC, Hive Writer executes the `LOAD DATA INPATH` statement to load data based on the mode specified by the writeMode parameter. `"hiveConfig": { "hiveCommand": "", "jdbcUrl": "", "username": "", "password": "" }` • Hive Writer allows you to write data to HDFS files by using an HDFS client. You can use the hiveConfig parameter to specify advanced settings for the HDFS client.	Yes	None

Configure Hive Writer by using the codeless UI

On the DataStudio page, double-click a data sync node, and perform the following operations on the node configuration tab that appears:

1. Configure the connections.

Configure the connections to the source and destination data stores for the sync node.

Parameter	Description
Connection	The datasource parameter in the preceding parameter description. Select a connection type, and then select a connection name that you have configured in DataWorks.
Table	The table parameter in the preceding parameter description.
Partition Key Column	The partition to which data is written. The last-level partition must be specified. Hive Writer can write data to only one partition.
Writing Rule	The writeMode parameter in the preceding parameter description.

2. Configure field mapping. It is equivalent to setting the column parameter in the preceding parameter description. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right.

GUI Element	Description
Map Fields with the Same Name	Click **Map Fields with the Same Name** to establish a mapping between fields with the same name. Note that the data types of the fields must match.
Map Fields in the Same Line	Click **Map Fields in the Same Line** to establish a mapping for fields in the same row. Note that the data types of the fields must match.
Delete All Mappings	Click **Delete All Mappings** to remove mappings that have been established.
Auto Layout	Click Auto Layout to sort the fields based on specified rules.

3. Configure channel control policies.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read data from and write data to data storage within the sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.

Parameter	Description
Resource Group	The resource group used for running the sync node. By default, the node runs on the default resource group. If resources are insufficient, you can add a custom resource group and run the sync node on the custom resource group. Set the resource group properly based on network conditions of the connections, resource group usage, and business importance.

Configure Hive Writer by using the code editor

In the following code, a node is configured to write data to Hive in JSON format.

```
{
  "type": "job",
  "steps": [
    {
      "stepType": "hive",
      "parameter": {
      },
      "name": "Reader",
      "category": "reader"
    },
    {
      "stepType": "hive",
      "parameter": {
        "partition": "year=a,month=b,day=c", // The partition to which data is written.
        "datasource": "hive_ha_shanghai", // The connection name.
        "table": "partitiontable2", // The name of the destination table.
        "column": [// The columns in the destination table to which data is written.
          "id",
          "name",
          "age"
        ],
        "writeMode": "append" // The write mode.
      },
      "name": "Writer",
      "category": "writer"
    }
  ],
  "version": "2.0",
  "order": {
    "hops": [
      {
```

```
            "from": "Reader",
            "to": "Writer"
          }
        ]
    },
    "setting": {
      "errorLimit": {
        "record": ""
      },
      "speed": {
        "throttle": false,
        "concurrent": 2
      }
    }
}
```

2.4.5.4.25. Configure Vertica Writer

Vertica is a column-oriented database using the MPP architecture. Vertica Writer allows you to write data to tables stored in Vertica databases. This topic describes how Vertica Writer works, its parameters, and how to configure it by using the code editor.

How it works

Vertica Writer connects to a remote Vertica database by using JDBC, and executes an `INSERT INTO` statement to write data to the Vertica database. Internally, data is submitted to the Vertica database in batches.

Vertica Writer is designed for ETL developers to import data from data warehouses to Vertica databases. Vertica Writer can also be used as a data migration tool by users such as DBAs.

Vertica Writer obtains data from a Data Integration reader, and generates the INSERT INTO statement based on your configurations.

- `INSERT INTO` : If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows.
- Data can be written only to tables stored in the primary Vertica database.

  > **Note** A sync node that uses Vertica Writer must have at least the permission to execute the `INSERT INTO` statement. Whether other permissions are required depends on the SQL statements specified in the preSql and postSql parameters when you configure the node.

- Vertica Writer does not support the writeMode parameter.
- Vertica Writer accesses a Vertica database by using the Vertica database driver. Confirm the compatibility between the driver version and your Vertica database. Vertica Writer uses the

following version of the Vertica database driver:

```
<dependency>
    <groupId>com.vertica</groupId>
    <artifactId>vertica-jdbc</artifactId>
    <version>7.1.2</version>
</dependency>
```

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be the same as the name of the added connection. You can add connections in the code editor.	Yes	None
jdbcUrl	The JDBC URL for connecting to the Vertica database. You do not need to set this parameter because the system automatically obtains the value from the connection parameter. • You can configure only one JDBC URL for a database. Vertica Writer cannot write data to a database with multiple primary databases. • The format must be in accordance with Vertica official specifications. You can also specify the information of the attachment facility. Example: `jdbc:vertica://127.0.0.1:3306/database` .	Yes	None
username	The username that you can use to connect to the database.	Yes	None
password	The password that you can use to connect to the database.	Yes	None
table	The names of the destination tables, which are described in a JSON array. Note You do not need to set this parameter because the system automatically obtains the value from the connection parameter.	Yes	None
column	The columns in the destination table to which data is written. Separate the columns with a comma (,), for example, `"column":["id","name","age"]` .	Yes	None
preSql	The SQL statement to execute before the sync node is run. Use `@table` to specify the name of the destination table in the SQL statement. When you execute this SQL statement, DataWorks replaces @table with the name of the destination table.	No	None

Parameter	Description	Required	Default value
postSql	The SQL statement to execute after the sync node is run.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Vertica database over the network, and increase the throughput. However, an excessively large value may lead to the OOM error during the data synchronization process.	No	*1,024*

Configure Vertica Writer by using the codeless UI

The codeless UI is not supported for Vertica Writer.

Configure Vertica Writer by using the code editor

In the following code, a node is configured to write data to a Vertica database.

```
{
   "type":"job",
   "version":"2.0",// The version number.
    "steps":[
      {
         "stepType":"stream",
         "parameter":{},
         "name":"Reader",
         "category":"reader"
      },
      {
         "stepType":"vertica",// The writer type.
          "parameter":{
            "datasource": "The connection name.",
            "username": "",
            "password": "",
            "column": [// The columns to which data is written.
                "id",
               "name"
            ],
            "connection": [
               {
                  "table": [// The name of the destination table.
                     "vertica_table"
                  ],
                  "jdbcUrl": "jdbc:vertica://ip:port/database"
```

```
            }
          ],
          "preSql": [ // The SQL statement to execute before the sync node is run.
            "delete from @table where db_id = -1"
          ],
          "postSql": [// The SQL statement to execute after the sync node is run.
            "update @table set db_modify_time = now() where db_id = 1"
          ]
        },
        "name":"Writer",
        "category":"writer"
      }
    ],
    "setting":{
      "errorLimit":{
        "record":"0"// The maximum number of dirty data records allowed.
      },
      "speed":{
        "throttle":false,// Specifies whether to enable bandwidth throttling. A value of false indicates t
hat the bandwidth is not throttled. A value of true indicates that the bandwidth is throttled. The maxi
mum transmission rate takes effect only if you set this parameter to true.
        "concurrent":1 // The maximum number of concurrent threads.
      }
    },
    "order":{
      "hops":[
        {
          "from":"Reader",
          "to":"Writer"
        }
      ]
    }
}
```

2.4.5.4.26. Configure Gbase8a Writer

This topic describes the implementation principle and parameter configurations of Gbase8a Writer.

Gbase8a Writer allows you to write data to tables stored in Gbase8a databases. At the underlying implementation level, Gbase8a Writer connects to a remote Gbase8a database through the JDBC Driver and runs the relevant SQL statements to write data to the Gbase8a database.

> **Note** You must configure a connection before configuring Gbase8a Writer.

Parameters

Parameter	Description	Required	Default value
datasource	The connection name. It must be identical to the name of the added connection. You can add connections in the code editor.	Yes	None
table	The name of the table to be synchronized.	Yes	None
writeMode	The write mode. Valid values: *insert into*, *on duplicate key update*, and *replace into*. • *insert into*: If a primary key conflict or unique index conflict occurs, data cannot be written to the conflicting rows and is regarded as dirty data. • *on duplicate key update*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, specified fields in original rows are updated. • *replace into*: If no primary key conflict or unique index conflict occurs, the action is the same as that of `insert into`. If a conflict occurs, original rows are deleted and new rows are inserted. That is, all fields of original rows are replaced.	No	*insert*
column	The columns in the destination table to which data is written. Separate the columns with a comma (,). Example: `"column": ["id", "name", "age"]`. Set the value to an asterisk (*) if data is written to all the columns in the destination table. That is, set the column parameter as follows: `"column":["*"]`.	Yes	None

Parameter	Description	Required	Default value
preSql	The SQL statement to run before the sync node is run. For example, you can clear outdated data before data synchronization. Currently, you can run only one SQL statement on the codeless user interface (UI), and multiple SQL statements in the code editor. Note If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
postSql	The SQL statement to run after the sync node is run. For example, you can add a timestamp after data synchronization. Currently, you can run only one SQL statement on the codeless UI, and multiple SQL statements in the code editor. Note If you specify multiple SQL statements in the code editor, the system does not guarantee that they are run in the same transaction.	No	None
batchSize	The number of data records to write at a time. Setting this parameter can greatly reduce the interactions between Data Integration and the Gbase8a database over the network, and increase the throughput. However, an excessively large value may lead to the out of memory (OOM) error during the data synchronization process.	No	*1024*

Configure Gbase8a Writer by using the codeless UI

Currently, the codeless UI is not supported for Gbase8a Writer.

Configure Gbase8a Writer by using the code editor

In the following code, a node is configured to write data to the Gbase8a database. For more information about the parameters, see the preceding parameter description.

```
{
  "type":"job",
  "version":"2.0",// The version number.
  "steps":[
    {
      "stepType":"stream",
      "parameter":{},
      "name":"Reader",
      "category":"reader"
    },
```

```
    {
      "stepType":"gbase8a",// The writer type.
      "parameter":{
        "postSql":[],// The SQL statement to run after the sync node is run.
        "datasource":"",// The connection name.
        "column":[// The columns to be synchronized.
          "id",
          "value"
        ],
        "writeMode":"insert",// The write mode.
        "batchSize":1024,// The number of data records to write at a time.
        "table":"",// The name of the table to be synchronized.
        "preSql":[]// The SQL statement to run before the sync node is run.
      },
      "name":"Writer",
      "category":"writer"
    }
  ],
  "setting":{
    "errorLimit":{// The maximum number of dirty data records allowed.
      "record":"0"
    },
    "speed":{
      "throttle":false,// Specifies whether to enable bandwidth throttling.
      "concurrent":1,// The maximum number of concurrent threads.
    }
  },
  "order":{
    "hops":[
      {
        "from":"Reader",
        "to":"Writer"
      }
    ]
  }
}
```

2.4.5.5. Optimize synchronization performance

This topic describes how to maximize the synchronization speed by adjusting the concurrency configuration, the difference between nodes that are configured with bandwidth throttling and those that are not, and precautions for custom resource groups.

Data Integration is a one-stop platform that supports real-time and offline data synchronization between any connections in any location and in any network environment. You can synchronize 10 TB of data between various types of cloud storage and local storage each day.

DataWorks provides excellent data transmission performance and supports data exchanges between more than 400 pairs of disparate connections. These features allow you to focus on the key issues on constructing big data solutions.

Factors affecting the speed of data synchronization

The factors that affect the speed of data synchronization are listed as follows:

- Source
  - Database performance: the performance of the CPU, memory module, SSD, network, and hard disk.
  - Concurrency: A high concurrency results in a heavy database workload.
  - Network: the bandwidth (throughput) and speed of the network. Generally, a database with better performance can support more concurrent nodes and a larger concurrency value can be set for sync nodes.
- Sync node
  - Synchronization speed: whether an upper limit is set for the synchronization speed.
  - Concurrency: a maximum number of concurrent threads to read data from the source and write data to destination data storage within a single sync node.
  - Nodes that are waiting for resources.
  - Bandwidth throttling: The bandwidth of a single thread is 1,048,576 bit/s. Timeout occurs when the business is sensitive to the network speed. We recommend that you set a smaller value.
  - Whether to create an index for query statements.
- Destination
  - Performance: the performance of the CPU, memory module, SSD, network, and hard disk.
  - Load: Excessive load in the destination database affects the write efficiency within the sync nodes.
  - Network: the bandwidth (throughput) and speed of the network.

You need to monitor and optimize the performance, load, and network of the source and destination databases. The following describes the optimal settings of a sync node.

Concurrency

You can configure the concurrency for a node on the codeless user interface (UI). The following is an example of how to configure the concurrency in the code editor:

```
"setting": {
    "speed": {
     "concurrent": 10
    }
  } }
```

Bandwidth throttling

By default, bandwidth throttling is disabled. In a sync node, data is synchronized at the maximum speed given the concurrency configured for the node. Considering that excessively fast synchronization may overstress the database and thus affect the production, Data Integration allows you to limit the synchronization speed and optimize the configuration as required. If bandwidth throttling is enabled, we recommend that you limit the maximum speed to 30 Mbit/s. The following is an example for configuring an upper limit for synchronization speed in the code editor, in which the transmission bandwidth is 1 Mbit/s:

```
"setting": {
    "speed": {
      "throttle": true // The bandwidth throttling is enabled.
      "mbps": 1, // The synchronization speed.
    }
  }
```

Note

- When the throttle parameter is set to false, throttling is disabled, and you do not need to configure the mbps parameter.
- The bandwidth value is a Data Integration metric and does not represent the actual network interface card (NIC) traffic. Generally, the NIC traffic is two to three times of the channel traffic, which depends on the serialization of the data storage system.
- A semi-structured file does not have shard keys. If multiple files exist, you can set the maximum job speed to increase the synchronization speed. However, the maximum job speed is limited by the number of files. For example, the maximum job speed limit is set to n Mbit/s for n files. If you set the limit to n+1 Mbit/s, the synchronization speed remains at n Mbit/s. If you set the limit to n-1 Mbit/s, the synchronization is performed at n-1 Mbit/s.
- A table can be partitioned according to the preset maximum job speed only when a maximum job speed and a shard key are configured for a relational database. Relational databases only support numeric shard keys, while Oracle databases support both numeric and string shard keys.

Scenarios of slow data synchronization

- Scenario 1: Resolve the issue that sync nodes to be run on the default resource group remain waiting for resources.

- Example

  When you test a sync node in DataWorks, the node remains waiting for resources and an internal system error occurs.

  For example, a sync node is configured to synchronize data from RDS to MaxCompute. The node has waited for about 800 seconds before it is run successfully. However, the log shows that the node runs for only 18 seconds and then stops. The sync node uses the default resource group. When you run other sync nodes, they also remain in the waiting state.

  The log is displayed as follows:

  ```
  2017-01-03 07:16:54 : State: 2(WAIT) | Total: 0R 0B | Speed: 0R/s 0B/s | Error: 0R 0B | Stage: 0.0%
  ```

- Solution

  The default resource group is not exclusively used by a single user. It is used by many projects concurrently, not just two or three nodes for a single user. If such resources are insufficient after you start to run a node, the node needs to wait for resources. In this case, the node is completed 800 seconds after you start running the node, but it only takes 10 seconds for the node to be executed.

  To improve the synchronization speed and reduce the waiting time, we recommend that you run sync nodes during off-peak hours. Typically, most sync nodes are run between 00:00 and 03:00.

- Scenario 2:

  Accelerate nodes that synchronize data from multiple source tables to the same destination table.

  - Example

    To synchronize data from tables of multiple data stores to a table, you configure multiple sync nodes to run in sequence. However, the synchronization takes a long time.

  - Solution

    To launch multiple concurrent nodes that write data to the same destination database, pay attention to the following points:

    - Ensure that the destination database can support the execution of all the concurrent nodes.
    - You can configure a sync node that synchronizes multiple source tables to the same destination table. Alternatively, you can configure multiple nodes to run concurrently in the same workflow.
    - If resources are insufficient, you can configure sync nodes to run during off-peak hours.

- Scenario 3:

  If no index is added when the WHERE clause is used, a full table scan slows down the data synchronization.

- Example

  SQL statement:

  ```
  select bid,inviter,uid,createTime from `relatives` where createTime>='2016-10-23 00:00:00'and rea
  teTime<'2016-10-24 00:00:00';
  ```

  Assume that the sync node started to run the preceding statement at 2016-10-25 11:01:24 and started to return results from 2016-10-25 11:11:05. It took a long time to finish the sync node.

- Cause

  When the WHERE clause is used for a query, the createTime column is not indexed, resulting in a full table scan.

- Solution

  We recommend that you use an indexed column or add an index to the column that you want to scan if you use the WHERE clause.

2.4.6. Full-database migration

2.4.6.1. Overview

This section describes the full-database migration feature in terms of its functions and limits.

Full-database migration is an easy-to-use tool that helps you to improve cost-efficiency. It can quickly upload all the tables in a MySQL database to MaxCompute at a time, saving time that is spent on creating batch tasks for initial data migration to the cloud.

For example, if a database contains 100 tables, you must configure 100 data synchronization tasks in a traditional way. With the full-database migration, you can upload all the tables at a time. However, an upload failure might occur due to the issues that involve the principles of designing database tables.

Task generation rules

After the configuration is completed, MaxCompute tables are created and data synchronization tasks are generated based on the selected tables to be synchronized.

The table names, field names, and field types of the MaxCompute tables are generated according to the advanced settings. If no advanced settings are configured, the structure of MaxCompute tables is identical to that of MySQL tables. The partition of these tables is pt, and its format is yyyymmdd.

The generated data synchronization tasks are daily scheduled tasks and run automatically on the early morning of the next day. The typical transmission rate is 1 Mbit/s, but it varies depending on the synchronization method and concurrency configurations. **To customize a data synchronization task, locate the task by choosing clone_database > Data Source Name > mysql2odps_table name**, and then specify its settings.

> **Note** We recommend that you perform smoke testing on a data synchronization task on the day when it is generated.To perform smoke testing, choose Administration Center > Task Management > project_etl_start > Upload Database > Data Source Name, find the synchronization task, right-click the task, and then test the task.

Limits

Full-database migration has the following limits due to the issues that involve the principles of designing database tables.

- Currently, only the full-database migration from a MySQL data source to MaxCompute is supported. We are working on support for full-database migration from a Hadoop or Hive data source to Oracle.
- Only the daily incremental and daily full upload modes are available.

  If you want to synchronize historical data at a time, this feature cannot meet your needs. We recommend that:

  - You configure daily tasks instead of synchronizing historical data at a time. You trace the historical data with the provided retrospective data import feature. This eliminates the need to run temporary SQL tasks to split data after all the historical data is synchronized.
  - To synchronize historical data at a time, configure a task on the task development page and click **Run**. Then, data is converted by using SQL statements. They are both one-time operations.

  If your daily incremental upload task uses a special business logic and cannot be identified by a date field, this feature cannot meet your needs. We provide the following suggestions:

  - The incremental data upload can be achieved by using two methods: binlog provided by the DTS product and the date field for data changes provided by databases.

    Currently, Data Integration supports the second method. Therefore, your database must contain the date field for data changes. The system determines whether your data is changed on the same day as the business date by using this field. If yes, all the changed data is synchronized.
  - To facilitate the incremental data uploading, we recommend that you include the gmt_create and gmt_modify fields when creating any database tables. Additionally, you can set the id field as the primary key to improve efficiency.
- Full-database migration supports batch upload and full upload modes.

  Batch upload is configured with time intervals. Currently, the connection pool protection feature for data sources is not supported, but will be available later.

  - To prevent overloads on the database, the full-database migration feature provides the batch upload mode. This mode enables you to upload tables in batches at a specified time interval and prevents compromised service functionality. We provide the following suggestions:
    - If you have master and slave databases, we recommend that you synchronize the data of the slave database.
    - In a batch upload task, each table has a database connection with a maximum transmission rate of 1 Mbit/s. For example, if you run a synchronization task for 100 tables at a time, 100 database connections are established. We recommend that you specify proper concurrency settings based on your business needs.
  - If you have special requirements for transmission efficiency, this feature cannot meet your needs. The maximum transmission of each generated tasks is 1 Mbit/s.
- Only the mapping of all table names, field names, and field types are supported.

  During the full-database migration process, MaxCompute tables are created automatically, where the partition field is pt, the field type is string, and the format is yyyymmdd.

  > **Note** When you select tables for synchronization, all fields must be synchronized and none of these fields can be edited.

2.4.6.2. Migrate a MySQL database

This topic describes how to migrate a MySQL database to MaxCompute.

The database migration feature improves efficiency and reduces costs. It can quickly upload all tables in a MySQL database to MaxCompute. For more information, see Overview.

Procedure

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, choose **Sync Resources > Connections**. On the page that appears, click **Add Connection**.
4. In the **Add Connection** dialog box that appears, select **MySQL**.
5. Add a MySQL connection named clone_database for database migration.
6. Click **Test Connection** and verify that the database can be accessed. Click **Complete**.
7. The added MySQL connection named clone_database appears in the connection list. Find the added connection and click **Migrate Database** in the Actions column.

   The database migration settings page consists of three functional modules.

Functional module	Description
Tables to migrate	This module lists all the tables in the MySQL connection named clone_database. Selected tables will be migrated.

Functional module	Description
Advanced Settings	You can configure the rules for converting the table name, column names, and data types.
Basic settings	You can select whether to synchronize full or incremental data, whether to upload data in one or more batches, and the synchronization efficiency. You can also view the migration progress and results.

8. Click **Advanced Settings** and configure conversion rules based on your needs. For example, you can add an ods_ prefix to the name of each MaxCompute table.
9. Specify basic settings. Set Sync Method to Synchronize Incremental Data Daily, and configure the incremental data to be determined based on the gmt_modified column. Data Integration will generate WHERE clauses based on the specified column and DataWorks scheduling parameters such as ${bdp.system.bizdate}.

   Data Integration reads data from MySQL tables by connecting to a remote MySQL database over JDBC and running SELECT statements. Data Integration uses standard SQL statements, and therefore you can configure WHERE clauses to filter data. The WHERE clause used in this example is provided as follows:

   ```
   STR_TO_DATE('${bdp.system.bizdate}', '%Y%m%d') <= gmt_modified AND gmt_modified < DATE_AD
   D(STR_TO_DATE('${bdp.system.bizdate}', '%Y%m%d'), interval 1 day)
   ```

   Select data upload in batches to protect the MySQL database from being overloaded. Let Data Integration start data synchronization for three tables every one hour from 00:00 each day.

   Click **Commit**. Then, you can view the migration progress and results of each table.
10. Find table a1 and click View Node to view the migration results.

You have configured a node for migrating a MySQL connection named clone_database to MaxCompute. This node is run based on the specified schedule, daily by default. You can also create retroactive node instances to transmit historical data. The database migration feature of **Data Integration** significantly simplifies the initial configurations for migrating your data to the cloud and reduces data migration costs.

You can view the migration success logs of table a1.

2.4.6.3. Migrate Oracle databases

This topic describes how to migrate an Oracle database to MaxCompute.

The database migration feature improves efficiency and reduces costs. It can quickly upload all tables in an Oracle database to MaxCompute. For more information, see Overview.

Procedure

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration** to go to the **Data Integration** page.
3. In the left-side navigation pane, choose **Sync Resources > Connections**. On the page that

appears, click **Add Connection** in the upper-right corner.

4. In the **Add Connection** dialog box that appears, select **Oracle**.
5. Add an Oracle connection named clone_databae for database migration.
6. Click **Test Connection** and verify that the database can be accessed. Click **Complete**.
7. The added Oracle connection named clone_databae appears in the connection list. Find the added connection and click **Migrate Database** in the Actions column.

   The database migration settings page consists of three functional modules.

Functional module	Description
Tables to migrate	This module lists all the tables in the Oracle connection named clone_databae. Selected tables will be migrated.
Advanced Settings	You can configure the rules for converting the table name, column names, and data types.
Basic settings	You can select whether to synchronize full or incremental data, whether to upload data in one or more batches, and the synchronization efficiency. You can also view the migration progress and results.

8. Click **Advanced Settings** and configure conversion rules based on your needs.
9. Set Sync Method to Synchronize All Data Daily.

   > **Note** If a date column exists in your table, you can select incremental migration and configure the incremental data to be determined based on the date column. Data Integration will generate WHERE clauses based on the specified column and DataWorks scheduling parameters such as ${bdp.system.bizdate}.

   Select data upload in batches to protect the Oracle database from being overloaded. Let Data Integration start data synchronization for three tables every one hour from 00:00 each day.

   Click **Commit**. Then, you can view the migration progress and results of each table.

10. Find a related table and click **View Node** to view the node details.

    You have configured a node for migrating an Oracle connection named clone_databae to MaxCompute. This node is run based on the specified schedule, daily by default. You can also create retroactive node instances to transmit historical data. The database migration feature of **Data Integration** significantly simplifies the initial configurations for migrating your data to the cloud and reduces data migration costs.

2.5. Data Analytics

2.5.1. Solution

The data analytics mode of DataWorks is upgraded so that you can group multiple workflows in a solution of a workspace.

Overview

DataWorks upgrades the data analytics mode to organize various types of nodes based on the business category. You can organize workflows to analyze data by business.

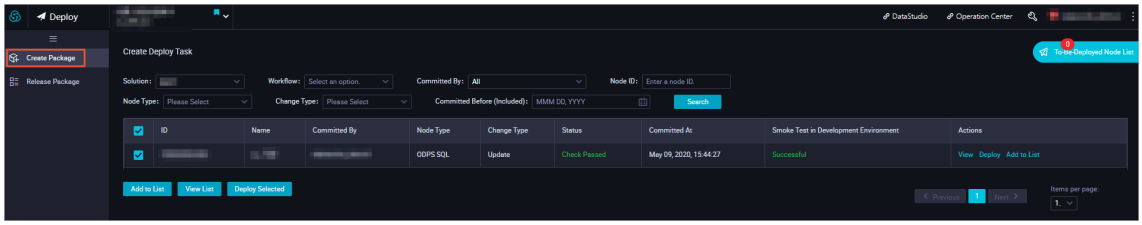
By using the data analytics mode that involves the **workspace**, **solution**, and **workflow**, DataWorks defines a new development process and improves user experience.

- A workspace is the basic organizational unit that manages the development and O&M permissions of users. The code of all nodes in a workspace can be collaboratively developed and managed by workspace members.
- A solution contains one or more workflows. It has the following advantages:
  - A solution can contain multiple workflows.
  - A workflow can be added to multiple solutions.
  - All solutions in a workspace can be collaboratively developed and managed by workspace members.
- A workflow is an abstract entity of business that enables you to develop data analytics code from a business perspective. A workflow can be added to multiple solutions. It has the following advantages:
  - Workflows facilitate business-oriented code development. Nodes in a workflow are organized by type. A hierarchical directory structure is supported. We recommend that you create a maximum of four levels of sub-directories. To create a sub-directory, right-click the target node type and select **Create Folder**.
  - You can view and optimize each workflow from a business perspective.
  - You can view each workflow on a dashboard to develop code with improved efficiency.
  - You can deploy and manage each workflow as a whole.

Develop a solution

If you double-click a solution in the left-side navigation pane, the left-side navigation pane only displays workflows in the solution. This prevents the development process from being affected by the code that is not related to the current solution in the workspace. To develop a solution, perform the following steps:

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over  and select **Solution**.
3. In the **Create Solution** dialog box, set **Solution Name** and **Description**, select a workflow from the **Workflows** drop-down list, and then click **Create**.
4. In the solution list, right-click the created solution and select **Solution Kanban**. On the solution dashboard that appears, you can view the selected workflows or modify the solution.
5. Move the pointer over the solution name. The  and  icons appear.
   - Click the  icon. The **Deploy** page appears. You can view the nodes to be deployed in the current solution.



> **Note** This icon is available only when the workspace is in standard mode.

- Click the icon to go to the **Cycle Instance** page under **Cycle Task Maintenance** in **Operation Center**. You can view recurring instances of all nodes in the current solution.

  In the left-side navigation pane, double-click the created solution. All the created workflows in the solution appear. You can click a workflow name to show the created nodes in it and perform operations on the nodes and the workflow.

  A workflow can be added to multiple solutions. After you develop a solution and add a workflow to the solution, other users can edit the workflow you referenced in their solutions for collaborative development.

2.5.2. SQL coding guidelines and specifications

This topic describes the basic guidelines and detailed specifications of SQL coding.

SQL coding guidelines

The SQL coding guidelines are as follows:

- Make sure that the code is comprehensive.
- Make sure that code lines are clear, neat, well-organized, and structured.
- Consider the optimal execution speed during SQL coding.
- Provide comments whenever necessary to enhance the readability of your code.
- The guidelines impose non-mandatory constraints on the coding behavior of developers. In practice, understandable deviations are allowed when developers obey general rules.
- Use lowercase letters for all keywords and reserved words. Keywords and reserved words include select, from, where, and, or, union, insert, delete, group, having, and count.
- In addition to keywords and reserved words, other code such as field names and table alias must be in lowercase.
- A unit of indentation contains four spaces. All indentations must be the integral multiple of an indentation unit. The code is aligned according to its hierarchy.
- The `select *` operation is prohibited. The column name must be specified for all operations.
- Matching opening and closing parentheses must be placed in the same column.

SQL coding specifications

The SQL coding specifications are as follows:

- Code header

The code header contains information such as the subject, description, author, and date. Reserve a line for change log and a title line so that later users can add change records. Each line can contain a maximum of 80 characters. The template is as follows:

```
-- MaxCompute(ODPS) SQL
--********************************************************************
-- ** Subject: Transaction
-- ** Description: Transaction refund analysis
-- ** Author: Youma
-- ** Created on: 20170616
-- ** Change log:
-- ** Modified on Modified by Content
-- yyyymmdd name comment
-- 20170831 Wuma Add a comment on the biz_type=1234 transaction
--********************************************************************
```

- Field arrangement
  - Use a line for each field that is selected for the SELECT statement.
  - Separate the first field from SELECT by one indentation unit.
  - Enter another field name in a separate line after two indentation units.
  - Place the comma (,) between two fields right before the second field.
  - Place the AS statement in the same line as the corresponding field. We recommend that you keep the AS statements of multiple fields in the same column.

```
select  channel_id            as channel_id
       ,trade_channel_desc    as trade_channel_desc
       ,trade_channel_edesc   as trade_channel_edesc
       ,inst_date             as inst_date
       ,trade_iswap           as trade_iswap
       ,channel_type          as channel_type
       ,channel_second_desc   as channel_second_desc
from   (
```

- Clause arrangement for an INSERT statement

  Arrange the clauses of an INSERT statement in the same line.

- Clause arrangement for a SELECT statement

  The clauses such as FROM, WHERE, GROUP BY, HAVING, ORDER BY, JOIN, and UNION in a SELECT statement must be arranged according to the following requirements:
  - Use a line for each clause.
  - Make sure that the clauses are left aligned with the SELECT statement.
  - Add two indentation units between the first word and the other code in a clause.
  - Keep the logical operators such as AND and OR in a WHERE clause left aligned with WHERE.

- If the length of a clause name exceeds two indentation units such as ORDER BY and GROUP BY, add a space between the clause name and its content.

```
select    trim(channel) channel
          ,min(id)      id
from      ods_trd_trade_base_dd
where     channel is not null
and       dt = ${tmp_uuuummdd}
and       trim(channel) <> ''
group by  trim(channel)
order by  trim(channel)
```

- Spacing before and after operators

  Keep one space before and one space after the arithmetic and logical operators and keep the operators in the same line, unless the clause contains more than 80 characters.

```
select      trim(channel) channel
            ,min(id)      id
from        ods_trd_trade_base_dd
where       channel is not null
and         dt = ${tmp_uuuummdd}
and         trim(channel) <> ''
group by    trim(channel)
order by    trim(channel)
```

- CASE statement

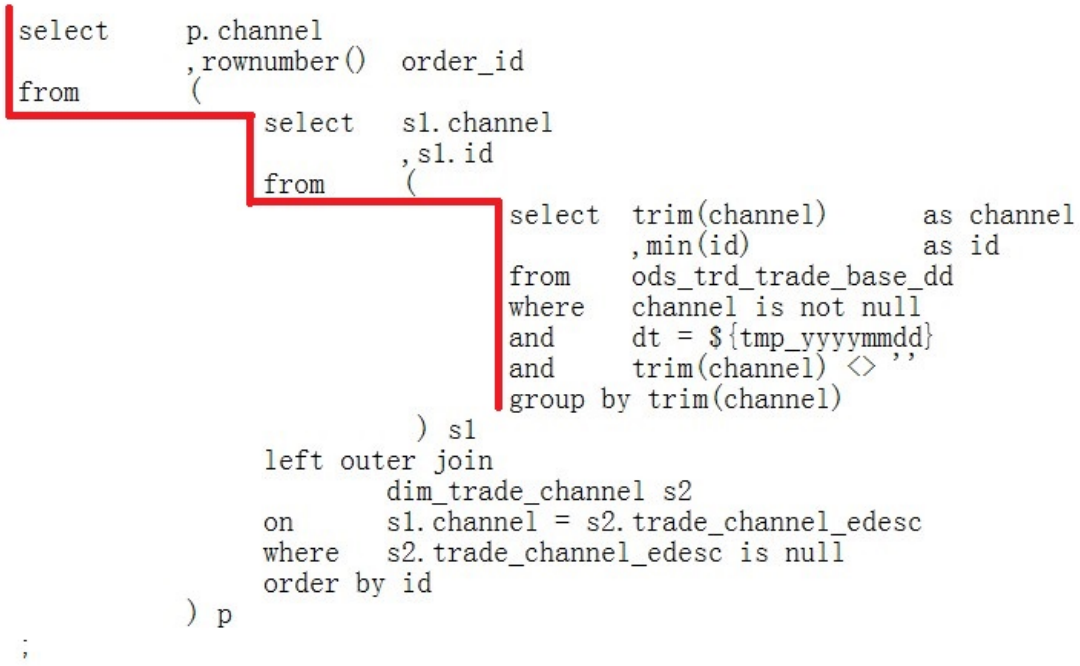
  The CASE statement can be used to determine the value of a field in a SELECT statement. Rules for writing CASE statements are as follows:

  - Place the WHEN clause in the same line as the CASE statement, with one indentation unit between them.
  - Keep a WHEN clause in one line whenever possible. If the statement is long, line breaks can be made.
  - A CASE statement must contain an ELSE clause. The ELSE clause must be aligned with the WHEN clause.

```
,case   when p1.trade_from = '3008' and p1.trade_email is null then 2
        when p1.trade_from = '4000' and p1.trade_email is null then 1
        when p9.trade_from_id is not null then p9.trade_from_id
 end                                as trade_from_id
,p1.trade_email                     as partner_id
```

- Nested query

Nested queries are often used in extract-transform-load (ETL) development of data warehouse systems. The following figure shows an example of a nested query.



```
select      p.channel
            ,rownumber()  order_id
from        (
                select    s1.channel
                          ,s1.id
                from      (
                                select  trim(channel)       as channel
                                        ,min(id)            as id
                                from    ods_trd_trade_base_dd
                                where   channel is not null
                                and     dt = ${tmp_yyyymmdd}
                                and     trim(channel) <> ''
                                group by trim(channel)
                           ) s1
                left outer join
                        dim_trade_channel s2
                on      s1.channel = s2.trade_channel_edesc
                where   s2.trade_channel_edesc is null
                order by id
            ) p
;
```

- Table alias
  - Once an alias is defined for a table in a SELECT statement, use the alias whenever you reference the table in the statement. Therefore, you must specify an alias for each table.
  - We recommend that you define the table aliases with simple characters, such as a, b, c, and d in sequence, and avoid using keywords.

- In the nested query, levels 1 to 4 of SQL statements are named part, segment, unit, and detail, which are abbreviated as P, S, U, and D. You can also use a, b, c, and d to represent levels 1 to 4.

  To differentiate multiple clauses at the same level, add numbers such as 1, 2, 3, and 4 next to the letters. Add comments to the table aliases as needed.

```
select     p.channel
           ,rownumber()  order_id
from       (
               select   s1.channel
                        ,s1.id
               from     (
                            select  trim(channel)      as channel
                                    ,min(id)           as id
                            from    ods_trd_trade_base_dd
                            where   channel is not null
                            and     dt = ${tmp_yyyymmdd}
                            and     trim(channel) <> ''
                            group by trim(channel)
                        ) s1
               left outer join
                       dim_trade_channel s2
               on      s1.channel = s2.trade_channel_edesc
               where   s2.trade_channel_edesc is null
               order by id
           ) p
;
```

- SQL comments
  - Add a comment for each SQL statement.
  - Use a separate line for the comment of each SQL statement and place the comment in front of the SQL statement.
  - Place the comment of a field right after the field.
  - Add comments for clauses that are difficult to understand.
  - Add comments for important code.
  - If a statement is long, we recommend that you add comments based on the purposes of each segment.
  - The description for a constant or variable is required. The comment on the valid value range is optional.

2.5.3. GUI elements

2.5.3.1. Overview

This topic describes the graphical user interface (GUI) elements on the DataStudio page and the configuration tab of an ODPS SQL node.

Log on to the DataWorks console. The **Data Analytics** page appears. You can double-click a created node to perform operations on the node configuration tab.

The following table describes the GUI elements.

No.	GUI element	Description
1	Show My Nodes Only icon	Click the icon to view your own nodes.
2	Search Code icon	Click the icon to search for a node or a code segment.
3	Create icon	Click the icon to create a solution, workflow, folder, node, table, resource, or function.
4	Refresh icon	Click the icon to refresh the directory tree in the left-side navigation pane.
5	Locate icon	Click the icon to find the current node in the left-side navigation pane.
6	Import icon	Click the icon to import local data to an online table. You must specify the encoding format. Note In a workspace of the standard mode, the local data is imported to a table in the development environment.
7	Filter icon	Click the icon to query nodes based on the specified filter conditions.
8	Save icon	Click the icon to save the code of the current node.
9	Save as Ad-Hoc Query Node icon	Click the icon to save the code of the current node in an ad-hoc query node. You can find the node on the Ad-Hoc Query tab.
10	Commit icon	Click the icon to commit the current node.
11	Commit and Unlock icon	Click the icon to commit and unlock the current node for editing.
12	Steal Lock icon	Click the icon to steal the lock of the current node and then edit it if you are not the owner of the node.
13	Run icon	Click the icon to run the code of the current node. You only need to assign values to variables in SQL statements once. The initial values are retained even if the node code changes.

No.	GUI element	Description
14	**Run with Arguments icon**	Click the icon to run the code of the current node with the configured parameters. You must manually assign values to variables in SQL statements each time you click this icon. The initial values are passed to the **Run with Arguments** feature, which replaces the initial values with the assigned values. For example, if the run date of a node is set to April 2, the node always runs on April 2 when you click the Run icon. After you click Run with Arguments icon and change the run date to April 3, the run date is updated. When you click the Run icon again, the node is run on April 3.
15	**Stop icon**	Click the icon to stop running the code of the current node.
16	**Reload icon**	Click the icon to reload the code of the current node. The code will be restored to the version last saved. Unsaved changes will be lost.
17	**Run Smoke Test icon**	Click the icon to test the code of the current node. A smoke test allows you to replace the values of scheduling parameters in the specified data timestamp with your simulated ones. This feature tests the effect of value changes for scheduling parameters. **Note** Each time after you modify the values of scheduling parameters, you must save and commit the modification before running the smoke test. Otherwise, the new values of scheduling parameters do not take effect.
18	**View Smoke Test Log icon**	Click the icon to view the runtime logs of the current script template.
19	**Format Code icon**	Click the icon to format the code to avoid excessively long code in a single line.
20	**Operation Center button**	Click the icon to go to Operation Center.
21	**Properties tab**	Click the tab to configure the properties such as the scheduling properties, parameters, and resource group for the current node.
22	**Lineage tab**	Click the tab to view the relationships between the current node and other nodes.
23	**Versions tab**	Click the tab to view the committed and deployed versions of the current node.

No.	GUI element	Description
24	Code Structure tab	Click the tab to view the code structure of the current node. If the code is excessively long, you can quickly find a code segment based on the key information in the structure.

2.5.3.2. Workflow Parameters

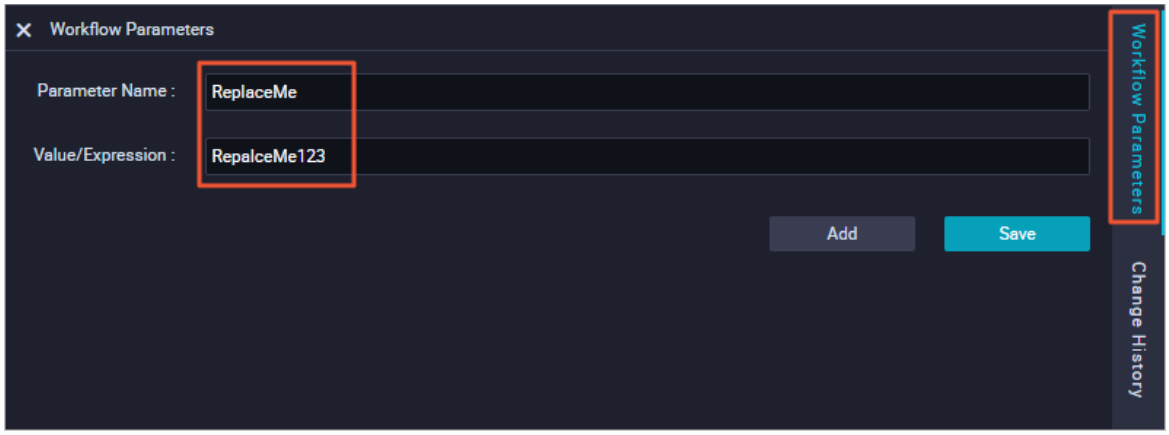
On the Workflow Parameters tab, you can assign a value to a variable or replace the value of a parameter for all nodes in the current workflow. This topic describes how to configure a workflow parameter by assuming that you want to replace the value of the ReplaceMe parameter with ReplaceMe123 in a manually triggered workflow.

Limits

- In manually triggered workflows, ODPS SQL nodes, Shell nodes, and sync nodes support global parameters. The format for specifying a global parameter varies based on the node type. For example, a global workflow parameter is specified as x=y1.
  - To configure the workflow parameter for an ODPS SQL node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter x=aaa in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with x=y1. You can use $x to reference the workflow parameter in the code.
  - To configure the workflow parameter for a Shell node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter $x in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with y1. You can use $1 to reference the workflow parameter in the code.
  - To configure the workflow parameter for a sync node, double-click the target node and click the **General** tab in the right-side navigation pane. On the **General** tab, enter -p"-Dx=aaa" in the Arguments field. When the node is run, x=aaa specified in the Arguments field is replaced with -p"-Dx=y1. You can use $x to reference the workflow parameter in the code.
- In auto triggered workflows, only ODPS SQL nodes support global parameters.
- Parameter names and values are case-sensitive.

Configure a workflow parameter

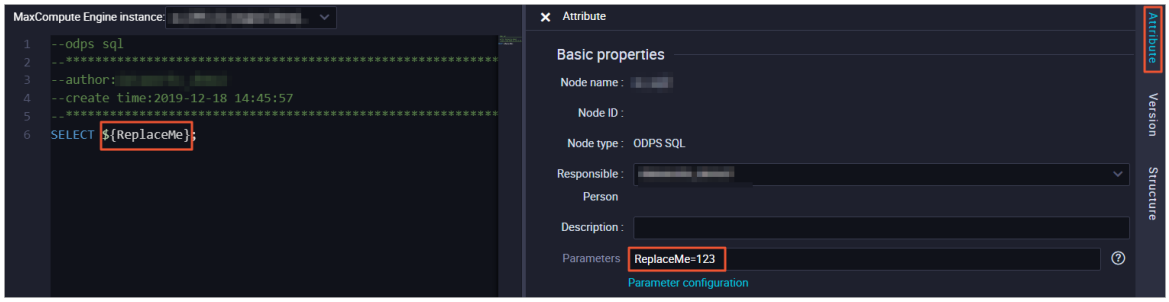
1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Manually Triggered Workflows**.
3. Double-click the target workflow to go to the workflow configuration tab.
4. In the right-side navigation pane, click the **Workflow Parameters** tab. In the Workflow Parameters pane, enter ReplaceMe in the **Parameter Name** field and ReplaceMe123 in the **Value/Expression** field.



5. Click in the toolbar.

Configure the workflow parameter for an ODPS SQL node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and choose **MaxCompute** > **Data Analytics** to show all the existing data analytics nodes. Double-click the target ODPS SQL node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter ReplaceMe=123 in the **Arguments** field.

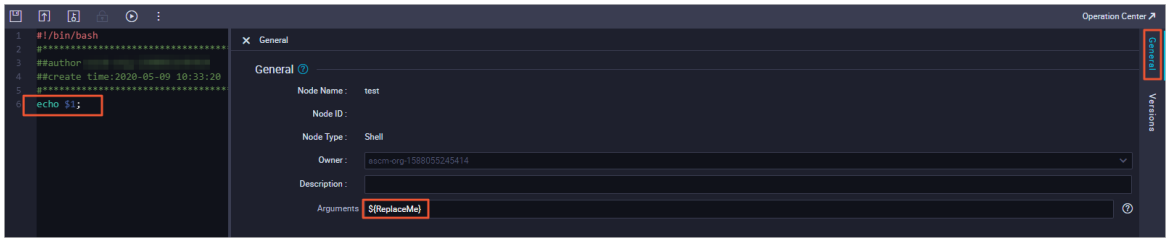


The workflow parameter is specified as ReplaceMe=ReplaceMe123. Therefore, the workflow parameter ReplaceMe is assigned the value ReplaceMe123 for this node when the workflow is run.

4. Click in the toolbar.

Configure the workflow parameter for a Shell node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and click **General** to show all the existing data analytics nodes. Double-click the target Shell node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter ${ReplaceMe} in the **Arguments** field.



> **Note** Make sure that you enter the parameter in the correct format.

4. Click in the toolbar.

Configure the workflow parameter for a sync node

1. On the **DataStudio** page, click **Manually Triggered Workflows** in the left-side navigation pane.
2. Find the target workflow and click **Data Integration** to show all the existing data integration nodes. Double-click the target sync node to go to the node configuration tab.
3. In the right-side navigation pane, click the **General** tab. In the General pane, enter -p"ReplaceMe=abc" in the **Arguments** field.

> **Note** Make sure that you enter the parameter in the correct format, namely, -p"-DParameter name=Parameter value".

4. Click in the toolbar.

Run the workflow to view the result

On the configuration tab of the workflow, click in the toolbar. In the Warning dialog box, click

Settings. In the **Runtime Parameters** dialog box, set Arguments to ReplaceMe. The value of the workflow parameter is replaced when the workflow is run.

You can use the following methods to view the value assigned to the workflow parameter for different types of nodes:

- Right-click the ODPS SQL node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the ODPS SQL node.
- Right-click the Shell node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the Shell node.
- Right-click the sync node and select **View Log**. Then, you can view the value assigned to the workflow parameter for the sync node.

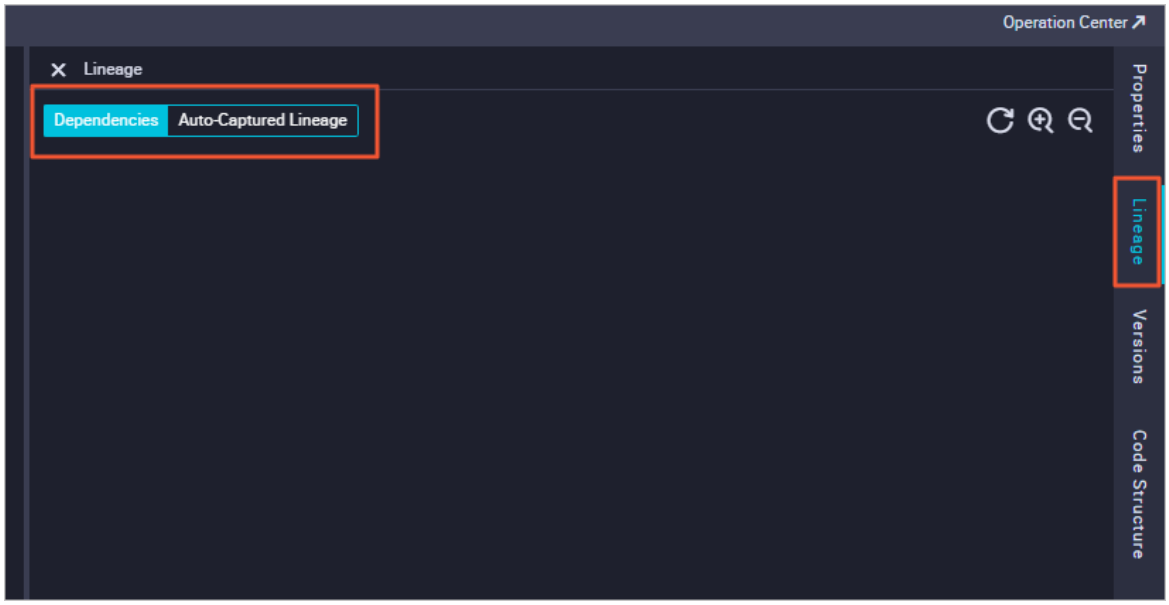
If you have not assigned a value to a workflow parameter on the **Workflow Parameters** tab for a manually triggered workflow, you must assign a value to the workflow parameter every time you run the workflow in the production environment.

2.5.3.3. Lineage

The Lineage tab displays the relationships between a node and other nodes. You can view the node dependencies and the lineage parsed from the code of the node.

Go to the Lineage tab

1. Log on to the DataWorks console.
2. Double-click the target node. For more information about how to create a node, see Create an ODPS SQL node.
3. On the node configuration tab that appears, click the **Lineage** tab in the right-side navigation pane.



On the **Lineage** tab, you can click **Dependencies** to view the dependencies or click **Auto-Captured Lineage** to view the lineage.

View the dependencies

You can check the node dependencies presented based on the current configuration. If the node dependencies fail to meet your expectations, you can reconfigure the node dependencies on the Properties tab.

View the auto-captured lineage

The lineage is parsed based on the code of the current node. For example, an ODPS SQL node contains the following SQL statements:

```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

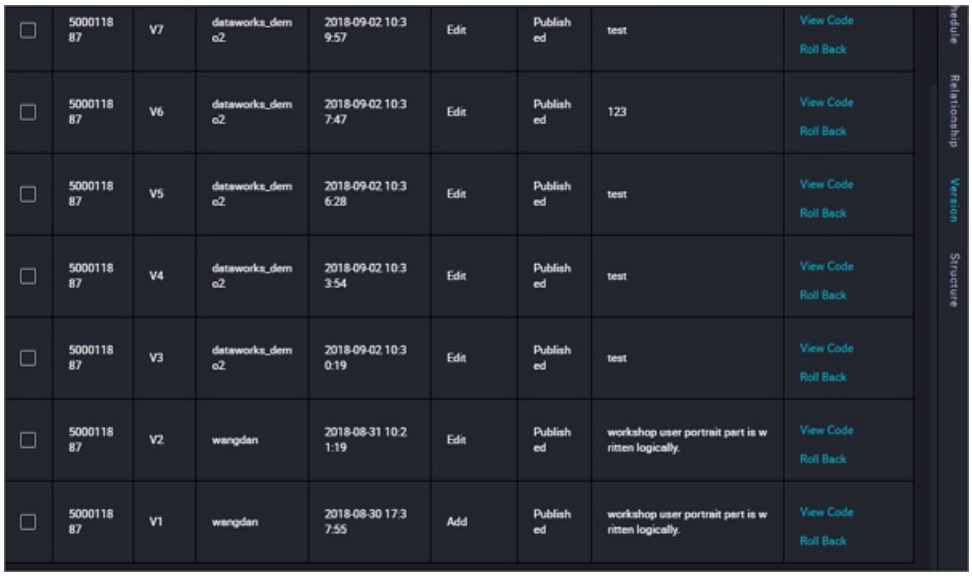
The following figure shows the lineage parsed from the preceding SQL statements. The results queried from the ods_log_info_d and ods_user_info_d tables are joined and then inserted into the dw_user_info_all_d table.

2.5.3.4. Versions

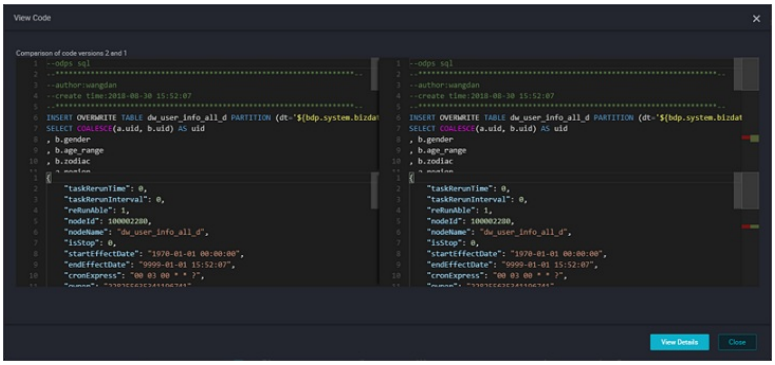
The Versions tab displays all committed and deployed versions of a node. You can view the historical versions and information about each version, including the user who committed the version, time when the version was committed, change type, status, and description.

> **Note** Only a committed node has the version information. Every time a node is committed, a version is generated and added to the **Versions** tab.

1. Log on to the DataWorks console.
2. On the **DataStudio** page, double-click the target node.
3. On the node configuration tab that appears, click the **Versions** tab in the right-side navigation pane. In the Versions pane, view the committed and deployed versions of the current node.



GUI element	Description
File ID	The ID of the node.
Versions	The version of the node. A version is generated each time the node is committed and deployed. V1 indicates version 1 and V2 indicates version 2. The version number is incremented by 1 each time.
Committed By	The user who committed the version.
Committed At	The time when the version was committed. If a version is committed and then deployed at a later time point, the value of this parameter is updated to the time when the version is deployed. By default, this column records the time when the version is last operated.
Change Type	The operation on the node. The value of this parameter is Create if the node is committed and deployed for the first time or Change if the node is modified, committed, and then deployed.

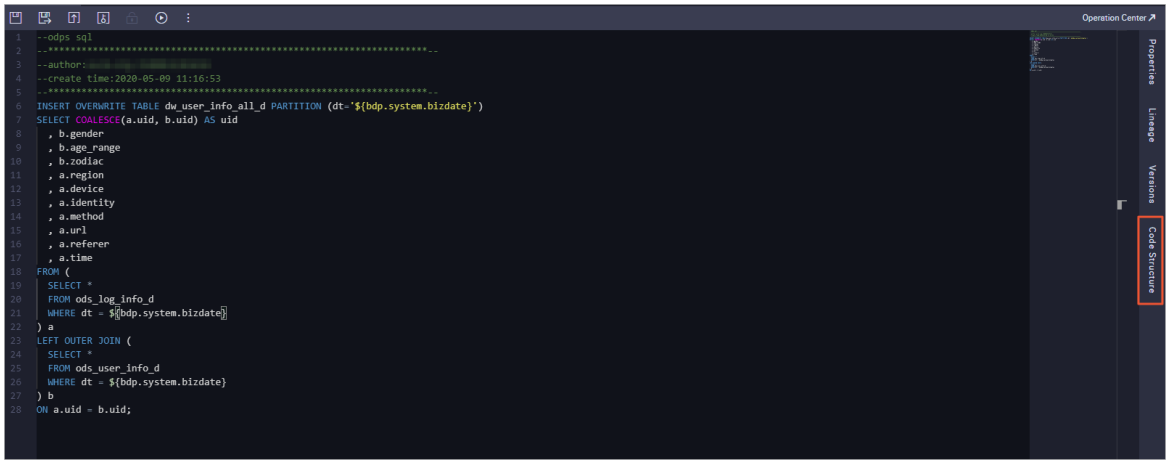
GUI element	Description
Status	The status of the version. Valid values: ○ **Yes**: The version is committed to the development environment but the related deployment task has not been created. The version has not been deployed in the production environment. ○ **Not Deployed**: The version is committed to the development environment and the deployment task is created. The version is pending for deployment. ○ **Deployed**: The version is committed to the development environment and deployed in the production environment.
Description	The change description of the version when it is committed. This description helps other users find the relevant version when they manage the node.
Actions	The actions that you can perform on the version. Two actions are available: **View Code** and **Roll Back**. ○ **View Code**: Click the button to view the code of the current version. ○ **Roll Back**: Click the button to roll back the node from the current version to the required version. After you roll back a node, you must commit and deploy it again.
Compare	Click the button to compare the code and properties between two selected versions.  Click **View Details**. On the details page that appears, you can view the changes in code and properties. **Note** You can only compare two versions and cannot compare one or more than two versions at a time.

2.5.3.5. Code Structure

The Code Structure tab displays the SQL code structure parsed from the code of a node. The code structure helps you view and modify the code.

1. Log on to the DataWorks console.

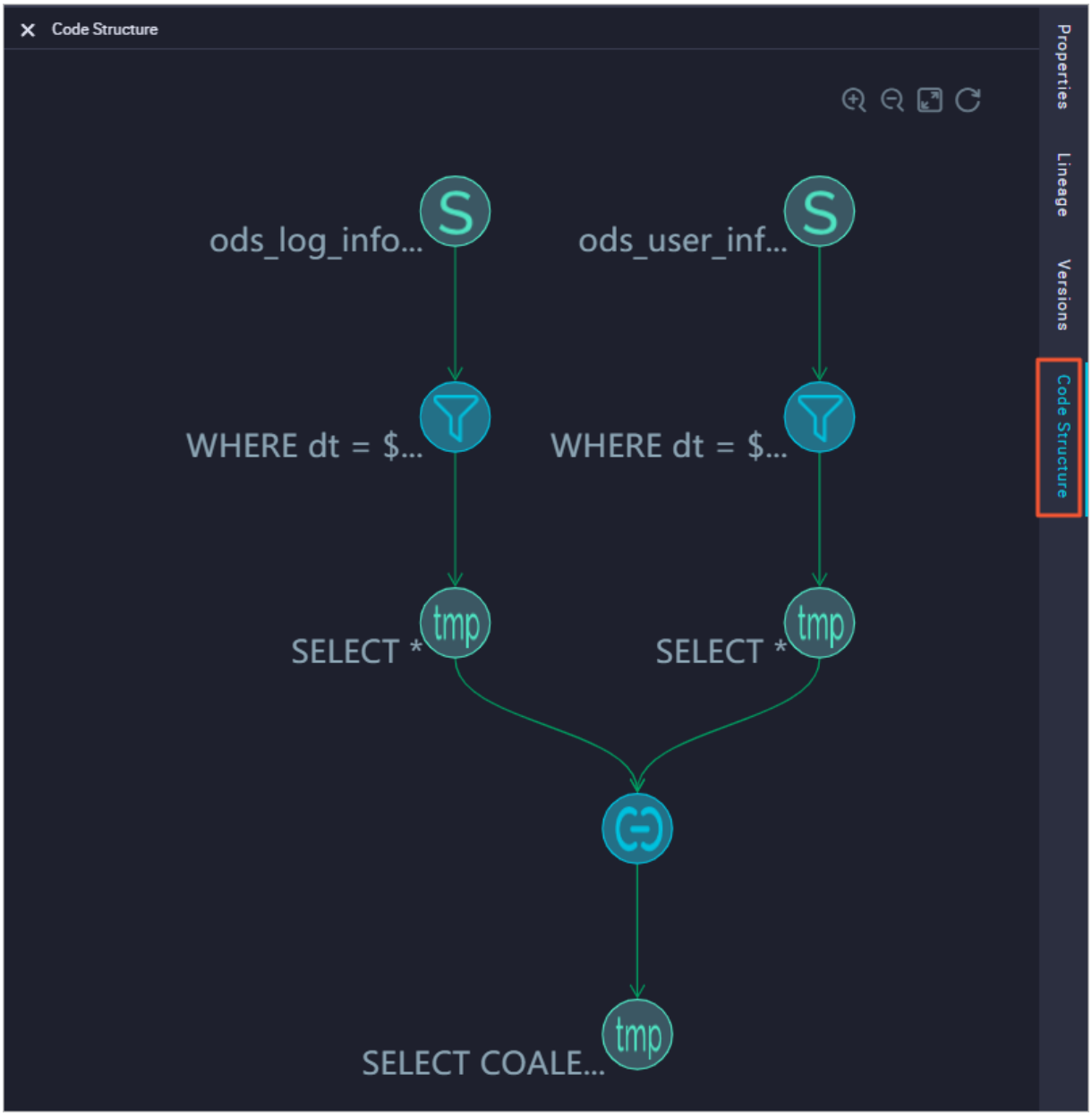
2. Double-click the ODPS SQL node whose code structure you want to view. For more information about how to create a node, see Create an ODPS SQL node.
3. On the node configuration tab that appears, click the **Code Structure** tab in the right-side navigation pane.



In this example, the ODPS SQL node contains the following SQL statement:

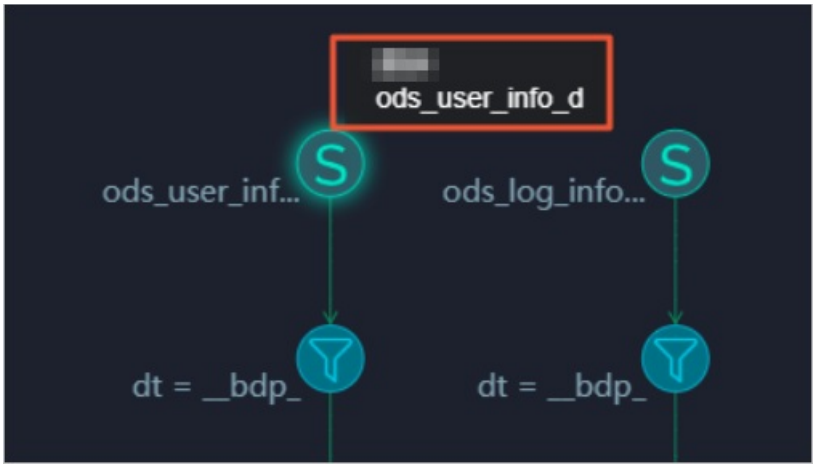
```
INSERT OVERWRITE TABLE dw_user_info_all_d PARTITION (dt='${bdp.system.bizdate}')
SELECT COALESCE(a.uid, b.uid) AS uid
  , b.gender
  , b.age_range
  , b.zodiac
  , a.region
  , a.device
  , a.identity
  , a.method
  , a.url
  , a.referer
  , a.time
FROM (
  SELECT *
  FROM ods_log_info_d
  WHERE dt = ${bdp.system.bizdate}
) a
LEFT OUTER JOIN (
  SELECT *
  FROM ods_user_info_d
  WHERE dt = ${bdp.system.bizdate}
) b
ON a.uid = b.uid;
```

The following figure shows the code structure parsed from the preceding SQL statement.

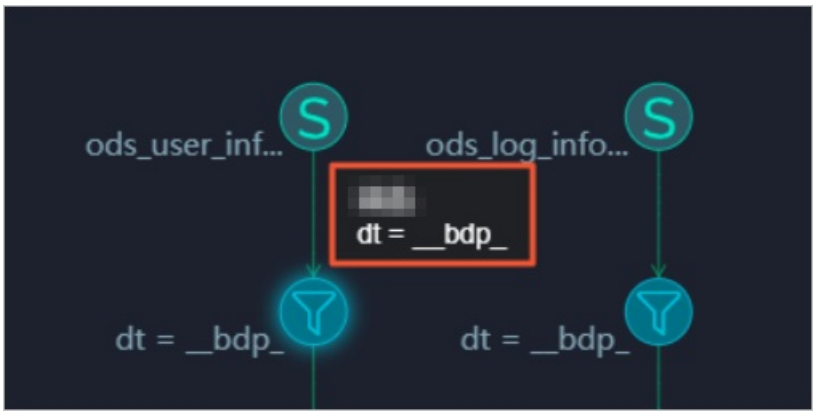


Move the pointer over a circle to view the description.

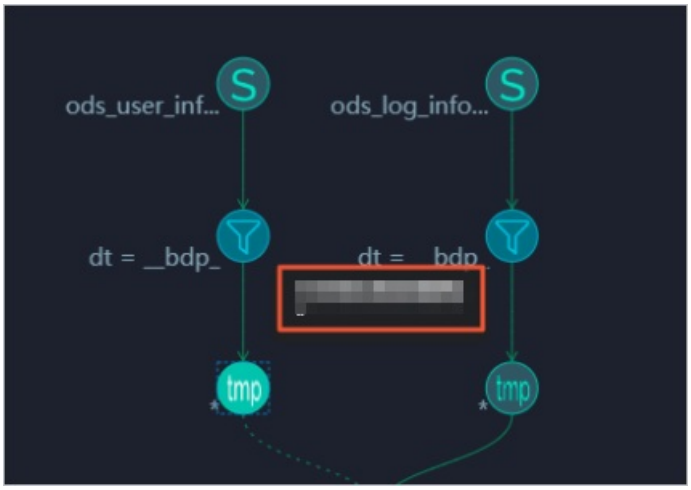
- **Source table**: the table to be queried.



- **Filter**: the condition for filtering the partitions in the table to be queried.

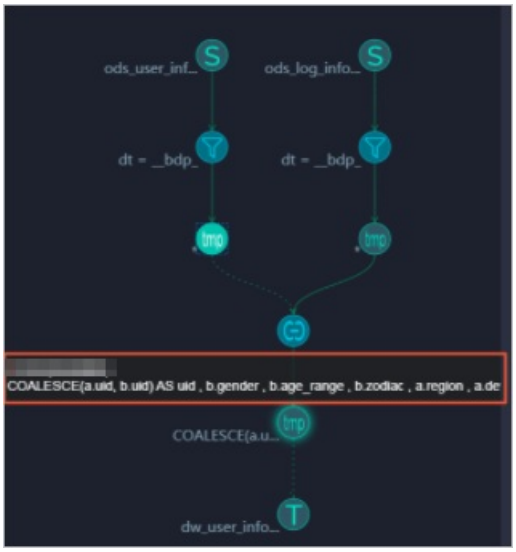


- **First intermediate table (view)**: the temporary table that stores the query results.

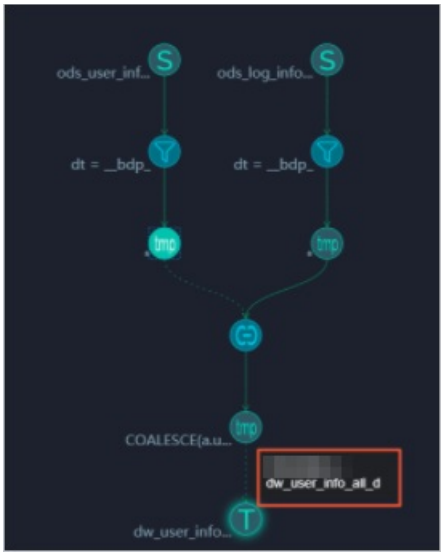


- **Join**: the operation for joining the query results.

- **Second intermediate table (view)**: the temporary table that stores the results of the JOIN operation. This temporary table can be stored for three days. After three days, this table is automatically deleted.



- **Destination table (insert)**: the destination table to which the query results are inserted by using an **INSERT OVERWRITE** statement.



2.5.4. Business flows

2.5.4.1. Overview

DataWorks organizes different types of nodes in a workflow by business category. This allows you to develop code by business.

DataWorks provides you with a dashboard for different types of nodes in each workflow. DataWorks also provides tools for you to optimize and manage nodes in each workflow. This promotes easy and intelligent development and management.

Workflow structure

A workspace supports multiple types of compute engines and multiple workflows. A workflow is a collection of various types of nodes that are closely associated with each other. DataWorks automatically generates a DAG so that you can view the workflow. A workflow supports the following types of nodes: data integration, data analytics, table, resource, function, and algorithm.

Each type of node has an independent folder. You can also create subfolders in each folder. To facilitate management, we recommend that you create a maximum of four levels of subfolders. If more than four subfolder levels are required, your workflow is too complex. We recommend that you split the workflow into two or more workflows and add the split workflows to one solution.

Create a workflow

1. Log on to the DataWorks console.
2. In the left-side navigation pane, click **Data Analytics**.
3. On the **Data Analytics** tab, right-click **Business Flow** and select **Create Workflow**.
4. In the **Create Workflow** dialog box, set **Workflow Name** and **Description**.

   > **Notice** The name of the workflow, which cannot exceed 128 characters in length.

5. Click **Create**.

Workflow nodes

A workflow consists of the following types of nodes:

- **Data integration**

  Click the target workflow and double-click **Data Integration** to view all data integration nodes of the workflow.

- **MaxCompute**

  The MaxCompute engine supports various data analytics nodes, such as ODPS SQL, SQL Snippet, ODPS Spark, PyODPS, ODPS Script, and ODPS MR nodes. You can also view and create tables, resources, and functions.

  - **Data analytics**

    Right-click Data Analytics under **MaxCompute** in the target workflow and select Create to create a data analytics node of a specific type.

  - **Table**

    Click the target workflow and choose **Create** > **Table** under **MaxCompute** to create a table. You can also view all the tables that are created in the current MaxCompute project.

  - **Resource**

    Click the target workflow, choose **Create** > **Resource** under **MaxCompute**, and then click a specific resource type to create a resource. You can also view all the resources that are created in the current MaxCompute project.

  - **Function**

    Click the target workflow and choose **Create** > **Function** under **MaxCompute** to create a function. You can also view all the functions that are created in the current MaxCompute project.

- **EMR**

  The E-MapReduce compute engine supports the following types of data analytics nodes: EMR Hive, EMR MR, EMR Spark, and EMR Spark SQL. You can also view and create E-MapReduce resources.

  > **Note** The **EMR** folder is available only after you create an E-MapReduce compute engine on the **Project Management** page.

  - **Data analytics**

    Click the target workflow, right-click **Data Analytics** under **EMR**, and then select Create to create a data analytics node of a specific type.

  - **Resource**

    Click the target workflow, right-click **Resource** under **EMR**, and then select Create to create a resource of a specific type. You can also view all the resources that are created in the current E-MapReduce compute engine.

- **Algorithm**

Click the target workflow, right-click **Algorithm**, and then choose Create > PAI Experiment to create a PAI Experiment node. You can also view all the PAI Experiment nodes that are created in the current workflow.

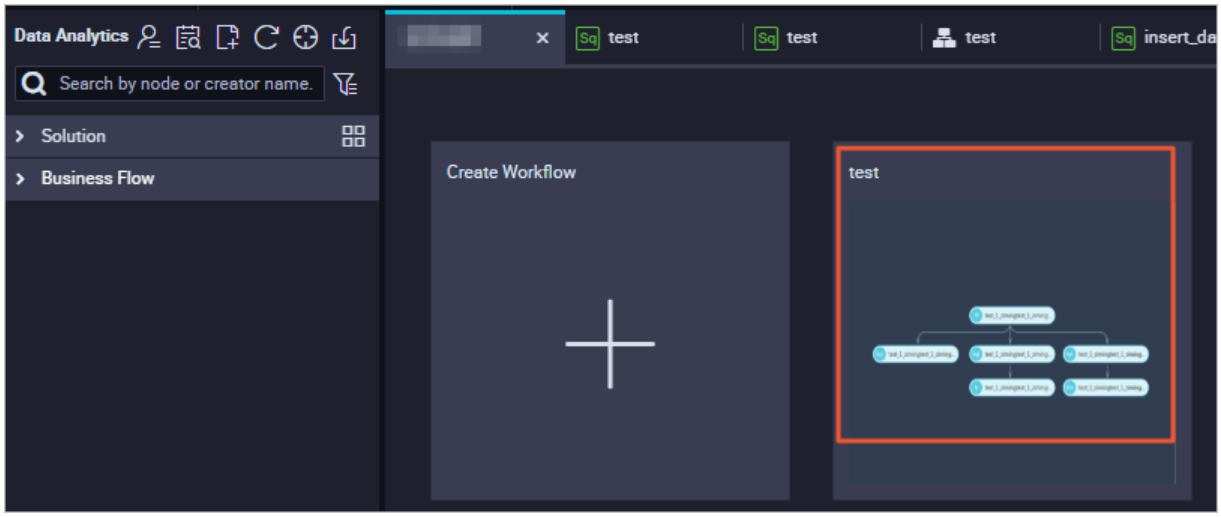
- **General**

  Click the target workflow, right-click **General**, and then select Create to create a node of a specific type.

View all workflows

On the **Data Analytics** tab, right-click **Business Flow** and select **All Workflows** to view all workflows that are created in the current workspace.

Click a workflow. The dashboard of the workflow appears.



View the dashboard for each node type

DataWorks provides a dashboard for each type of nodes in a workflow. On the dashboard, each node is presented by a card that offers operation and optimization suggestions, so that you can intelligently manage nodes.

For example, the card of each data analytics node provides two indicators to show whether baseline-based monitoring and event notification are enabled for the node. This allows you to understand the status of each node.

You can double-click a folder in a workflow to view the dashboard of the selected node type.

Commit a workflow

1. Go to the dashboard of a workflow and click in the toolbar.
2. In the **Commit** dialog box, select the nodes to be committed, set **Description**, and then select **Ignore I/O Inconsistency Alerts**.
3. Click **Commit**.

> **Note** If a node has been committed but the node code is not modified, the node cannot be selected again. In this case, you can enter your comments on the node and click **Commit**. The property changes of the node are automatically committed.

2.5.4.2. Create and reference a node group

This topic describes how to create and reference a node group.

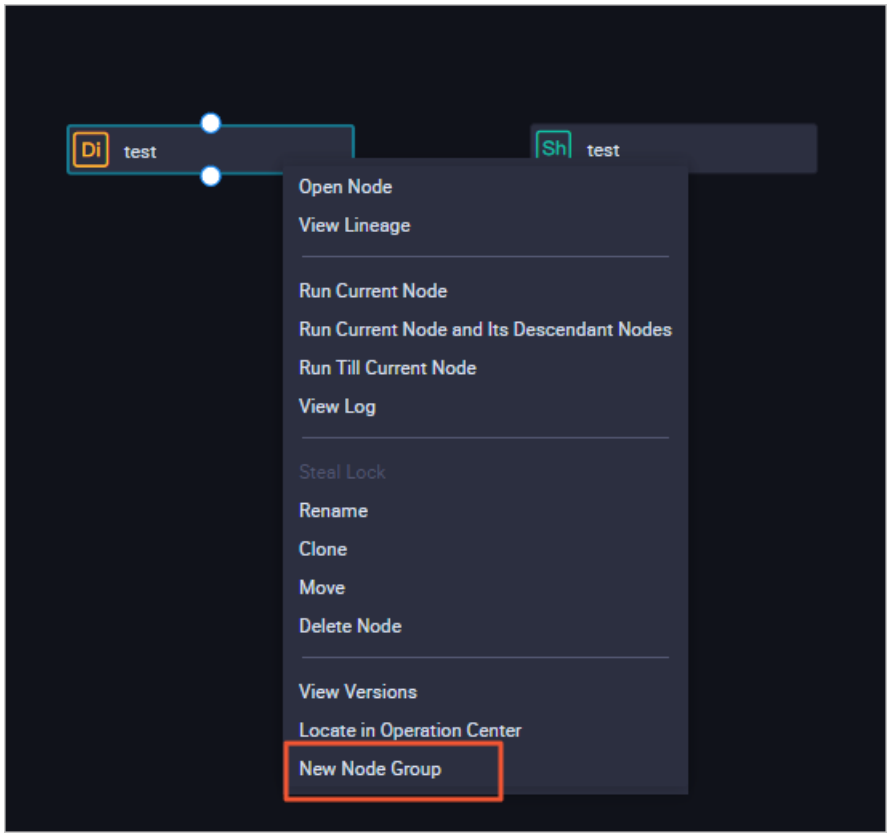
Context

You can group several nodes that are frequently reused together as a node group. The configuration of each node remains unchanged after the nodes are added to a node group. Later, you can directly reference the node group to reuse these nodes.

Create a node group

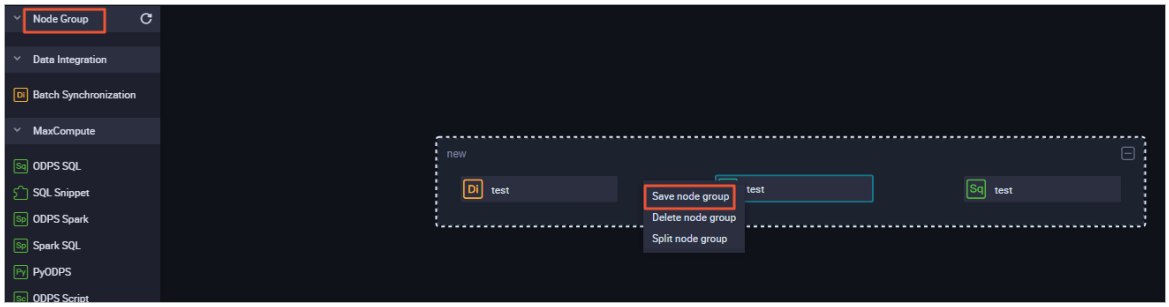
1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, create a workflow. For more information, see Create a workflow.
3. Go to the dashboard of the created workflow. Click in the upper-right corner and drag a box to select the target nodes to be included in a node group.

4. Right-click any node among the selected nodes and select **New Node Group**.



5. In the **New Node Group** dialog box, enter a name in the **Name** field and click **OK**.

6. Right-click the node group and select **Save node group**. In the dialog box that appears, click OK. Then, you can view the created node group in the **Node Group** section.



Menu item	Description
Save node group	Save the node group. The node group that you have created appears in the Node Group section only after you click **Save node group**. A node group that is not saved cannot be referenced in other workflows.
Delete node group	Delete the node group. Click **Delete node group** to delete all nodes in the selected node group.
Split node group	Dismiss the node group. After the node group is dismissed, the selected nodes no longer form a node group in the workflow. However, the node group still exists in the Node Group section.

> **Note** If the created node group contains a PAI Experiment node, create a PAI experiment in another workflow to reference the node group. If the created node group contains a branch node, add digits to the value in the **Associated Node Output** parameter.

Reference a node group

You can directly drag a node group to another workflow to reference the node group in the workflow. The dependencies among the nodes in the node group remain unchanged.

You can run the workflow or commit and deploy the workflow. Then, go to **Operation Center** to view the running result.

2.5.5. Node types

2.5.5.1. Data Integration

2.5.5.1.1. Create a batch sync node

Batch sync nodes support various types of data stores, including MaxCompute, MySQL, DRDS, SQL Server, PostgreSQL, Oracle, MongoDB, Db2, Table Store, OSS, FTP, HBase, LogHub, HDFS, and Stream.

Context

When you enter a table name, a drop-down list appears, displaying all matched tables. Only exact match is supported. Therefore, you must enter a complete table name. Tables are labeled as unsupported if they are not supported by batch sync nodes.

If you move the pointer over a table in the list, the details of the table appear, including the database, IP address, and owner of the table. After you select a table, the column information is automatically entered. You can add, move, and delete columns.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **Data Integration > Batch Synchronization**. Alternatively, you can click a workflow in the Business Flow section, right-click **Data Integration**, and then choose **Create > Batch Synchronization**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Configure the batch sync node. For more information, see Overview.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.2. MaxCompute

2.5.5.2.1. Create an ODPS SQL node

Using the SQL-like syntax, ODPS SQL nodes can process terabytes of data in distributed processing scenarios that do not require real-time processing.

Context

Generally, it takes a long time from preparing to committing a job. You can use ODPS SQL nodes to process thousands to tens of thousands of transactions. ODPS SQL nodes are online analytical processing (OLAP) applications designed to deal with large amounts of data.

Limits

- You cannot use SET statements, USE statements, or SQL alias statements independently in the code of an ODPS SQL node. They must be executed together with other SQL statements. For example, you can use a SET statement together with a CREATE TABLE statement.

```
set a=b;
create table name(id string);
```

- You cannot add comments to statements containing keywords, including SET statements, USE statements, and SQL alias statements, in the code of an ODPS SQL node. For example, the following comment is not allowed:

```
create table name(id string);
set a=b; // Comment.
create table name1(id string);
```

- The running of an ODPS SQL node during workflow development and the scheduled running of an ODPS SQL node have the following differences:
  - Running during workflow development: combines all the statements containing keywords, including SET statements, USE statements, and SQL alias statements, in the node code and executes them before executing other SQL statements.
  - Scheduled running: executes all SQL statements in sequence.

```
set a=b;
create table name1(id string);
set c=d;
create table name2(id string);
```

The following table shows the differences between the two running modes for the preceding SQL statements.

SQL statement	Running during workflow development	Scheduled running
First SQL statement	`set a=b;` `set c=d;` `create table name1(id string);`	`set a=b;` `create table name1(id string);`
Second SQL statement	`set a=b;` `set c=d;` `create table name2(id string);`	`set c=d;` `create table name2(id string);`

- You must specify a scheduling parameter in the format of key=value. Do not add any spaces before or after the equation mark (=). Examples:

```
time = {yyyymmdd hh:mm:ss} // Incorrect format.
a =b // Incorrect format.
```

- If you use keywords such as bizdate and date as scheduling parameters, you must specify the values in the format of yyyymmdd. If you want to use other time formats, do not use the preceding keywords as scheduling parameters. Example:

```
bizdate=201908 // Incorrect format.
```

- You can only use statements starting with SELECT, READ, or WITH to query the result data for a node during the workflow development. Otherwise, no results are returned.
- Separate multiple SQL statements with semicolons (;) and place them in different lines.
  - Incorrect example

```
create table1;create table2
```

  - Correct example

```
create table1;
create table2;
```

Create an ODPS SQL node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over  and choose **MaxCompute > ODPS SQL**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS SQL**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Edit the code of the ODPS SQL node.Edit the code of the ODPS SQL node. The code must conform to the syntax. The following example creates a table, inserts data to the table, and queries data in the table:

i. Create a table named test1.

```
CREATE TABLE IF NOT EXISTS test1
( id BIGINT COMMENT '',
 name STRING COMMENT '',
 age BIGINT COMMENT '',
 sex STRING COMMENT '');
```

ii. Insert data to the table.

```
INSERT INTO test1 VALUES (1,'Zhang San',43,'Male');
INSERT INTO test1 VALUES (1,'Li Si',32,'Male');
INSERT INTO test1 VALUES (1,'Chen Xia',27,'Female');
INSERT INTO test1 VALUES (1,'Wang Wu',24,'Male');
INSERT INTO test1 VALUES (1,'Ma Jing',35,'Female');
INSERT INTO test1 VALUES (1,'Zhao Qian',22,'Female');
INSERT INTO test1 VALUES (1,'Zhou Zhuang',55,'Male');
```

iii. Query data in the table.

```
select * from test1;
```

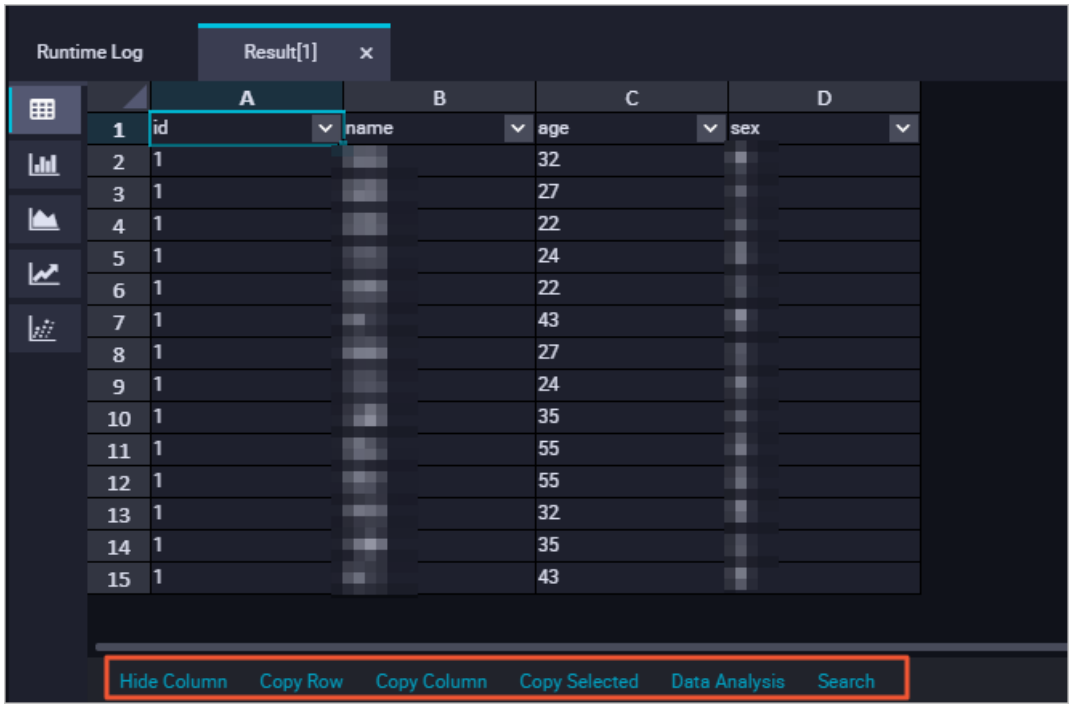
iv. After you enter the preceding SQL statements in the code editor, click  in the toolbar.

DataWorks executes your SQL statements from top to bottom and displays logs.The `INSERT INTO` statement may result in unexpected data duplication. Although DataWorks does not re-execute the `INSERT INTO` statement, it may rerun corresponding nodes. We recommend that you avoid using the `INSERT INTO` statement. When DataWorks executes the INSERT INTO statement, the following information appears in logs:

```
The INSERT INTO statement in SQL may cause repeated data insertion. Although SQL-level retries have been revoked for the INSERT INTO statement, task level retries may still happen. We recommend that you avoid the use of the INSERT INTO statement.
If you continue to use INSERT INTO statements, we deem that you are aware of the associated risks and are willing to take the consequences of potential data duplication.
```

v. View the query result.DataWorks displays the query result in a workbook.

You can view or manage the query result in the workbook, or copy the query result to a local Excel file.



Action	Description
Hide Column	Select one or more columns and click **Hide Column** at the bottom to hide the selected columns.
Copy Row	Select one or more rows and click **Copy Row** at the bottom to copy the selected rows.
Copy Column	Select one or more columns and click **Copy Column** at the bottom to copy the selected columns.
Copy Selected	Select one or more cells in the workbook and click **Copy Selected** at the bottom to copy the selected cells.
Data Analysis	Click **Data Analysis** at the bottom to go to the workbook editing page.
Search	Click **Search** at the bottom to search for data in the workbook. After you click the button, a search box appears in the upper-right corner of the Results tab.

6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click in the toolbar.

ii. In the **Commit Node** dialog box, ignore the alert on mismatch between the input and output that you set with those detected in code lineage analysis, enter your comments in the **Description** field, and then select **I confirm to proceed with the commission.**

iii. Click **OK.**

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the ODPS SQL node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.2.2. Create an SQL Snippet node

SQL script templates are SQL templates that involve multiple input and output parameters. Each SQL script template involves one or more source tables. You can use an SQL script template to filter, join, or aggregate data in source tables.

Context

When a new version is released for a script template, you can decide whether to upgrade the version of the script template used in your nodes to the latest version.

The script template upgrade mechanism allows developers to continuously upgrade script template versions. This mechanism enhances the process execution efficiency and optimizes the business performance.

For example, User A uses V1.0 of a script template that belongs to User B. Then, User B releases V2.0 for the script template. User A receives a notification of the new version. After User A compares the code of the two versions, User A can decide whether to upgrade the script template to the latest version.

To upgrade an SQL script template, click **Update Code** and check whether the parameter configuration of the SQL script template is valid in the new version. Set parameters for the SQL script template of the new version based on the version description. Then, save the node and commit it for deployment.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **MaxCompute > SQL Snippet.**

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > SQL Snippet.**

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, set **Node Name** and **Location.**

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the node configuration tab, select a script template from the **Snippet** drop-down list. To improve development efficiency, you can create data analytics nodes by using the script templates provided by workspace members and tenants.
   - The script templates provided by members of the current workspace are available on the **Workspace-Specific** tab.
   - The script templates provided by tenants are available on the **Public** tab.
6. Click the **Parameters** tab in the right-side navigation pane and set parameters for the SQL script template.
7. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
8. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click  in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.
9. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.2.3. Create an ODPS Spark node

DataWorks supports ODPS Spark nodes. This topic uses the JAR resource type as an example to describe how to create and configure an ODPS Spark node.

Create and upload a resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over  and choose **MaxCompute** > **Resource** > **JAR**. Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create** > **Resource** > **JAR**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.
3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.
4. Click **Upload** and select the target file to upload.
5. Click **OK**.

Create an ODPS Spark node

1. On the **Data Analytics** tab, move the pointer over and choose **MaxCompute > ODPS Spark**.

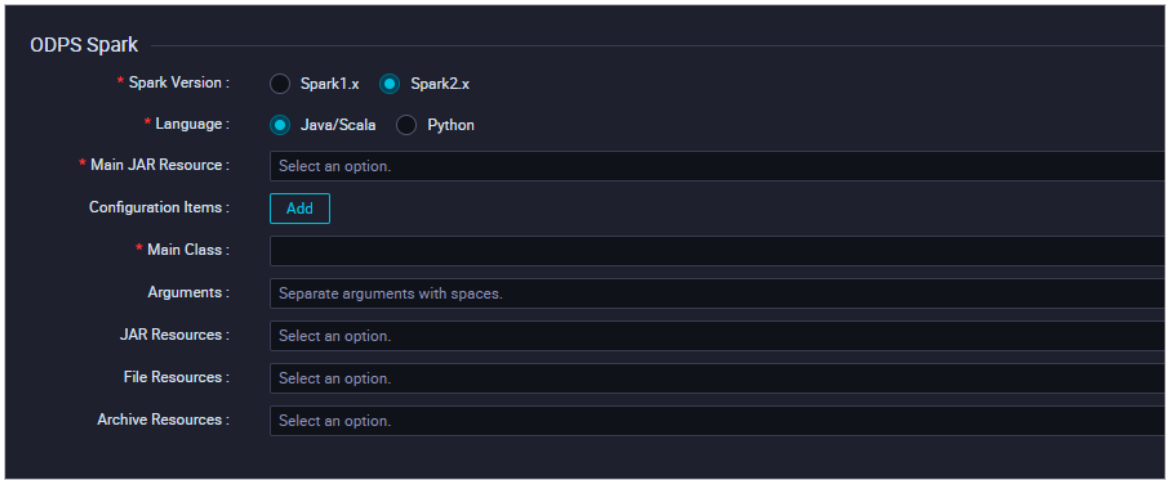
   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS Spark**.

2. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

3. Click **Commit**.

4. On the node configuration tab, set the parameters.



You can set **Spark Version** and **Language** as needed. The parameters vary with the value of the **Language** parameter. You can set the parameters as prompted.

The following table describes the parameters that appear after you set the Language parameter to **Java/Scala**.

Parameter	Description
Spark Version	The Spark version of the node. Valid values: **Spark1.x** and **Spark2.x**.
Language	The programming language of the node. Valid values: **Java/Scala** and **Python**. Select **Java/Scala**.
Main JAR Resource	The main JAR resource referenced by the node. Select a JAR resource that you uploaded from the drop-down list.
Configuration Items	The configuration items of the node. Click **Add** and set key and value to add a configuration item.
Main Class	The class name of the node.
Arguments	The parameter used to assign a value to a variable in the code during node scheduling. Separate multiple parameters with spaces.

Parameter	Description
JAR Resources	The JAR resource referenced by the node. Select a JAR resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded JAR resources based on the resource type.
File Resources	The file resource referenced by the node. Select a file resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded file resources based on the resource type.
Archive Resources	The archive resource referenced by the node. Select an archive resource that you uploaded from the drop-down list. The ODPS Spark node automatically finds the uploaded archive resources based on the resource type. Only compressed resources appear.

5. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   Notice You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click  in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.
7. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.2.4. Create a PyODPS node

DataWorks supports PyODPS nodes, which are integrated with the Python SDK of MaxCompute. You can edit Python code in PyODPS nodes of DataWorks to process data in MaxCompute.

Context

You can also use the Python SDK of MaxCompute to process data in MaxCompute.

Note
- The Python version of PyODPS nodes is 2.7.
- Each PyODPS node can process a maximum of 50 MB data and can occupy a maximum of 1 GB memory. Otherwise, DataWorks terminates the PyODPS node. Avoid writing too much data processing code for a PyODPS node.

PyODPS nodes are designed to use the Python SDK of MaxCompute. If you want to run pure Python code, you can create a Shell node to run the Python scripts uploaded to DataWorks.

Create a PyODPS node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over  and choose **MaxCompute > PyODPS**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > PyODPS**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Edit the code of the PyODPS node on the node configuration tab.
   i. Use the MaxCompute entry.Each PyODPS node includes the global variable odps or o, which is the MaxCompute entry. Therefore, you do not need to manually specify the MaxCompute entry.

   ```
   print(odps.exist_table('PyODPS_iris'))
   ```

   ii. Run SQL statements. In PyODPS nodes, you can execute MaxCompute SQL statements to query data and obtain the query results. You can use the execute_sql or run_sql method to run MaxCompute job instances.

   To execute statements that are not directly compatible with the MaxCompute console, you can use some methods. For example, you cannot directly execute statements other than DDL and DML in the MaxCompute console.

   To execute a GRANT or REVOKE statement, use the run_security_query method. To run a PAI command, use the run_xflow or execute_xflow method.

   ```
   o.execute_sql('select * from dual')  # Execute the statement in synchronous mode. Other node
   s are blocked until the SQL statement is executed.
   instance = o.run_sql('select * from dual') # Execute the statement in asynchronous mode.
   print(instance.get_logview_address()) # Obtain the Logview URL of an instance.
   instance.wait_for_success() # Other nodes are blocked until the SQL statement is executed.
   ```

iii. Set runtime parameters. You can use the hints parameter to set the runtime parameters. The type of the hints parameter is DICT.

```
o.execute_sql('select * from PyODPS_iris', hints={'odps.sql.mapper.split.size': 16})
```

If you set the sql.settings parameter for the global configuration, you must set the runtime parameters each time you run the code.

```
from odps import options
options.sql.settings = {'odps.sql.mapper.split.size': 16}
o.execute_sql('select * from PyODPS_iris')  # The hints parameter is automatically set based o
n the global configuration.
```

iv. Obtain SQL query results. You can use the open_reader method to obtain query results in the following scenarios:

- The SQL statement returns structured data.

```
with o.execute_sql('select * from dual').open_reader() as reader:
for record in reader:  # Process each record.
```

- SQL statements such as DESC are executed. In this case, you can use the reader.raw property to obtain raw query results.

```
with o.execute_sql('desc dual').open_reader() as reader:
print(reader.raw)
```

> **Note** If you use a custom time variable, you must fix the variable to a time. PyODPS nodes do not support relative time variables.

6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click  in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

Built-in modules for PyODPS nodes

A PyODPS node contains the following built-in modules:

- setuptools

- cython
- psutil
- pytz
- dateutil
- requests
- pyDes
- numpy
- pandas
- scipy
- scikit_learn
- greenlet
- six
- Other built-in modules in Python 2.7, such as smtplib

2.5.5.2.5. Create an ODPS Script node

You can create an ODPS Script node to develop an SQL script by using the SQL engine provided by MaxCompute V2.0.

Context

The ODPS Script node allows DataWorks to compile the SQL script as a whole, instead of compiling the SQL statements in the script one by one. In this way, the SQL script is committed and run as a whole. This guarantees that an execution plan is only queued and executed once, making full use of MaxCompute computing resources.

Create an ODPS Script node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **MaxCompute > ODPS Script**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > ODPS Script**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Edit the SQL script of the ODPS Script node as required.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.

7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click  in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

SQL syntax and limits for ODPS Script nodes

Write SQL statements based on your business logic in a way similar to that of using a common programming language. You do not need to consider how to organize the SQL statements.

```
-- SET statements
set odps.sql.type.system.odps2=true;
[set odps.stage.reducer.num=***;]
[...]
-- DDL statements
create table table1 xxx;
[create table table2 xxx;]
[...]
-- DML statements
@var1 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM table3
  [WHERE where_condition];
@var2 := SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM table4
  [WHERE where_condition];
@var3 := SELECT [ALL | DISTINCT] var1.select_expr, var2.select_expr, ...
  FROM @var1 join @var2 on ... ;
INSERT OVERWRITE|INTO TABLE [PARTITION (partcol1=val1, partcol2=val2 ...)]
  SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM @var3;
[@var4 := SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var1
  UNION ALL | UNION
  SELECT [ALL | DISTINCT] var1.select_expr, var.select_expr, ... FROM @var2;
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  AS
  SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM var4;]
```

SQL syntax and limits for ODPS Script nodes

- ODPS Script nodes support SET statements, DML statements, and some DDL statements. The DDL statements used to return data, such as DESC and SHOW statements, are not supported.
- A complete script consists of SET statements, DDL statements, and DML statements in sequence. You can write one or more statements of each type, or even skip a type without writing any statements of that type. However, you cannot mix different types of statements together. You must strictly follow the sequence of SET statements > DDL statements > DML statements.
- The at signs (@) residing before some statements indicate that these statements are connected by using variables.
- A script supports only one statement that returns data, such as an independent SELECT statement. If multiple such statements are provided, an error occurs. We recommend that you do not use SELECT statements in a script.

- A script supports only one `CREATE TABLE AS` statement, which must be the last statement. We recommend that you put CREATE TABLE statements and INSERT statements in different sections to separate them.
- If one statement in a script fails, the whole script fails.
- A job is generated to process data only after all the input data is prepared for a script.
- If a script writes data to a table and then reads the table, an error occurs. For example, an error occurs for the following statements:

```
insert overwrite table src2 select * from src where key > 0;
@a := select * from src2;
select * from @a;
```

To avoid the error, modify the statements to the following:

```
@a := select * from src where key > 0;
insert overwrite table src2 select * from @a;
select * from @a;
```

Sample script:

```
create table if not exists dest(key string , value bigint) partitioned by (d string);
create table if not exists dest2(key string,value bigint ) partitioned by (d string);
@a := select * from src where value >0;
@b := select * from src2 where key is not null;
@c := select * from src3 where value is not null;
@d := select a.key,b.value from @a left outer join @b on a.key=b.key and b.value>0;
@e := select a.key,c.value from @a inner join @c on a.key=c.key;
@f := select * from @d union select * from @e union select * from @a;
insert overwrite table dest partition (d='20171111') select * from @f;
@g := select e.key,c.value  from @e join @c on e.key=c.key;
insert overwrite table dest2 partition (d='20171111') SELECT * from @g;
```

Scenarios of ODPS Script nodes

- You can use an ODPS Script node to rewrite a single statement with nested subqueries, or a script that must be split into multiple statements due to its complexity.
- Data from different data stores may be prepared at different time points, and the time difference may be large. For example, the data from one data store can be prepared at 01:00, whereas that from another data store can be prepared at 07:00. In this case, table variables are not suitable for connecting statements. You can use an ODPS Script node to combine the statements to a script.

2.5.5.2.6. Create an ODPS MR node

MaxCompute supports the MapReduce API. You can create and commit ODPS MR nodes that call the Java API operations of MapReduce to develop MapReduce programs for processing data in MaxCompute.

Context

Before you create ODPS MR nodes, you must upload, commit, and then deploy required resources.

Create a resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **MaxCompute > Resource > JAR**. Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > JAR**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.

   > **Note**
   > - The resource name can be different from the name of the uploaded file.
   > - Convention for naming resources: A resource name can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive and must be 1 to 128 characters in length. A JAR resource name must end with .jar. A Python resource name must end with .py.

4. Click **Upload** and select the target file to upload.
5. Click **OK**.
6. Click in the toolbar to commit the resource to the development environment.

Create an ODPS MR node

1. On the **Data Analytics** tab, find the target workflow, right-click **MaxCompute**, and then choose **Create > ODPS MR**.
2. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

3. Click **Commit**.
4. Edit the ODPS MR node.

```
-- Create an input table.
CREATE TABLE if not exists jingyan_wc_in (key STRING, value STRING);
-- Create an output table.
CREATE TABLE if not exists jingyan_wc_out (key STRING, cnt BIGINT);
   --- Create the dual table.
   drop table if exists dual;
   create table dual(id bigint); -- Create the dual table if no dual table exists in the current works
pace and initialize the table.
   --- Initialize the dual table.
   insert overwrite table dual select count(*)from dual;
   --- Insert the sample data to the wc_in table.
   insert overwrite table jingyan_wc_in select * from (
   select 'project','val_pro' from dual
   union all
   select 'problem','val_pro' from dual
   union all
   select 'package','val_a' from dual
   union all
   select 'pad','val_a' from dual
    ) b;
-- Reference the uploaded JAR package. You can find the JAR package in the resource list, right-cli
ck the JAR resource, and select Insert Resource Path.
--@resource_reference{"mapreduce-examples.jar"}
jar -resources mapreduce-examples.jar -classpath ./mapreduce-examples.jar com.aliyun.odps.ma
pred.open.example.WordCount jingyan_wc_in jingyan_wc_out
```

Pay attention to the following information when you write the code:

- `--@resource_reference` : references a resource. Find the target resource, right-click it, and then select **Insert Resource Path** to generate the reference statement.
- `-resources` : the name of the referenced JAR resource.
- `-classpath` : the path of the JAR resource. You can enter ./Resource name because the resource has been referenced.
- `com.aliyun.odps.mapred.open.example.WordCount` : the main class in the JAR resource to be called during node running. It must be the same as the main class name in the JAR resource.
- `jingyan_wc_in` : the name of the input table of the ODPS MR node. The input table is created in the preceding code.
- `jingyan_wc_out` : the name of the output table of the ODPS MR node. The output table is created in the preceding code.
- If you use multiple JAR resources in a single ODPS MR node, separate the resource paths

with commas (,), for example, `-classpath ./xxxx1.jar,./xxxx2.jar` .

5. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click  in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.
7. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.2.7. Create a MaxCompute table

This topic describes how to create a MaxCompute table.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over  and choose **MaxCompute > Table**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Table**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.
3. In the **Create Table** dialog box, set **Table Name** and click **Commit**.

   > **Notice** A table name can be up to 64 characters in length. The table name must start with a letter and cannot contain Chinese or special characters.
4. On the table configuration tab that appears, set the parameters in the **General** section.

Parameter or button	Description
Display Name	The display name of the table.

Parameter or button	Description
Level 1 Folder	The name of the level-1 folder where the table resides. Note Level-1 and level-2 folders only show the table locations in DataWorks so that you can better manage tables.
Level 2 Folder	The name of the level-2 folder where the table resides.
Create Folder	Goes to the **Folder Management** tab. On this tab, you can create level-1 and level-2 folders for tables.
Description	The description of the table.

5. Create a table. Use one of the following methods to create a table:
   - Create a table by using DDL statements.

     Click **DDL Statement** in the top navigation bar. In the dialog box that appears, enter the statements for creating a table.

     After you finish editing the statements, click **Generate Table Schema**. Information is automatically entered in the General, Physical Model, and Schema sections.
   - Create a table on the graphical user interface (GUI).

     If DDL statements are inappropriate for you to create a table, try to use the GUI. The following table describes the relevant parameters for creating a table on the GUI.

Section	Parameter or button	Description
Physical Model	**Partitioning**	Specifies whether the table is partitioned. Valid values: **Partitioned Table** and **Non-Partitioned Table**.
	Time-to-Live	The time-to-live of data in MaxCompute. If you select this check box, you must enter a number in the **TTL** field. If the table or partition is stored for more than the specified number of days, data that has not been updated is cleared.
	Table Level	The level of the table. Generally, tables are divided into operation data store (ODS), common data model (CDM), and application data service (ADS) levels. You can specify a custom level name.

Section	Parameter or button	Description
	Categories	The category of the table. Tables are categorized into basic services, advanced services, and other services. You can specify a custom category name. If you want to create a table category or level, click **Create Level** to go to the **Level Management** tab. **Note** Categories are designed only for your management convenience and do not involve underlying implementation.
	Table Type	The type of the table. Default value: **Internal Table**.
Schema	Field Name	The name of the field. The name can contain letters, digits, and underscores (_).
	Display Name	The display name of the field.
	Data Type	The data type of the field.
	Definition or Maximum Value Length	The maximum value length of a field. You can set a maximum value length only for fields of the DECIMAL, VARCHAR, ARRAY, MAP, and STRUCT types.
	Description	The description of the field.
	Primary Key Field	Specifies whether the field serves as the primary key or part of a composite primary key.
	Create Field	Adds a field to the table.
	Delete Field	Deletes a field from the table. **Note** If you delete a field from an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Move Up	Adjusts the field sequence of the table. If you adjust the sequence of fields in an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Move Down	The description is the same as that of the **Move Up** operation.

Section	Parameter or button	Description
	Add	Adds a partition to the table. If you add a partition to an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Delete	Deletes a partition from the table. If you delete a partition from an existing table, DataWorks requests you to delete the table and create another table with the same name. This operation is forbidden in the production environment.
	Actions	Commits a partition or deletes a field.
Partition Field Design Note This section is available only when Partitioning under Physical Model is set to Partitioned Table.	**Add**	Adds a partition field.
	Data Type	The data type of the partition field. We recommend that you use the STRING type for all partition fields.
	Length	The maximum length of the partition field. You can set the maximum length only for fields of the VARCHAR-type.
	Description	The description of the partition field.
	Partition Column Date Format	The format of the date partition. If the partition field is a date, although the data type may be STRING, select or enter a date format, such as *yyyy mmmdd* or *yyyy-mm-dd*.
	Partition Column Date Granularity	The granularity of the date partition. The granularities can be second, minute, hour, day, month, quarter, and year. You can enter a partition granularity as required. If you want to specify multiple partition granularities, note that a greater granularity corresponds to a higher partition level. For example, three partitions whose granularities are day, hour, and month, respectively, are available. Multi-level partitions are in the hierarchical order of level-1 partition (month), level-2 partition (day), and level-3 partition (hour).

6. Click **Commit in Development Environment** and **Commit to Production Environment** in sequence.If you are using a workspace in basic mode, you only need to click **Commit to Production Environment**.

Button	Description

Button	Description
Load from Development Environment	If the table has been committed to the development environment, the button is clickable. After you click the button, the information about the table you create in the development environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Commit in Development Environment	Before you click the button, make sure that you have filled in all required parameters on the table configuration tab. Do not click the button if any parameters are not specified.
Load from Production Environment	After you click the button, the information about the table that is committed to the production environment overwrites the table information on the current page. **Note** This feature is supported only for MaxCompute tables.
Commit to Production Environment	After you click the button, the table is created in the workspace of the production environment.

What's next

After the table is created, you can query the table data and modify or delete the table. For more information, see Manage tables.

2.5.5.2.8. Create, reference, and download resources

This topic describes how to create, reference, and download JAR and Python resources.

Context

If your code or function requires resource files such as.jar files, you can upload resources to your workspace and reference them.

If the existing built-in functions do not meet your requirements, DataWorks allows you to create user-defined functions (UDFs) and customize processing logic. You can upload the required JAR packages to your workspace so that you can reference them when you create UDFs.

Note

- You can view built-in functions on the **Built-In Functions** tab. For more information, see View built-in functions.
- You can view the UDFs that you have committed or deployed on the **MaxCompute Functions** tab.

The resources that you can upload to MaxCompute include text files, MaxCompute tables, Python code, and compressed packages in the .zip, .tgz, .tar.gz, .tar, and .jar formats. You can read or use these resources when you run UDFs or MapReduce.

MaxCompute provides API operations for you to read and use resources. The following types of MaxCompute resources are available:

- **Python**: the Python code you have written. You can use Python code to register Python UDFs.
- **JAR**: the compiled Java JAR packages.
- **Archive**: the compressed files that can be identified by the file name extension. Supported file types include .zip, .tgz, .tar.gz, .tar, and .jar.
- **File**: files in the .zip, .so, or .jar format.

JAR resources and **file** resources have the following differences:

- To create a JAR resource, write Java code in the offline Java environment, compress the code to a JAR package, and upload the package as a JAR resource to DataWorks.
- To create a file resource that is smaller than or equal to 500 KB in size, you can create and edit it in the DataWorks console.
- To create a file resource that is larger than 500 KB in size, select **Large File (more than 500 KB)** and click Upload to upload the file.

> **Note** Each resource file to be uploaded in the DataWorks console cannot exceed 30 MB.

Create a JAR resource

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **MaxCompute > Resource > JAR**. Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create > Resource > Python**.

   > **Notice** The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

3. In the **Create Resource** dialog box, set **Resource Name** and **Location**.

   > **Note**
   > - The resource name can be different from the name of the uploaded file.
   > - A resource name can contain letters, digits, underscores (_), and periods (.), and is not case-sensitive. It must be 1 to 128 characters in length. A JAR resource name must end with .jar, and a Python resource name must end with .py.

4. Click **Upload** and select the target file to upload.
5. Click **OK**.
6. Click in the toolbar to commit the resource to the development environment.

Create a Python resource and register a UDF

1. Create a Python resource.
   i. On the **Data Analytics** tab, move the pointer over and choose **MaxCompute** > **Resource** > **Python**.Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create** > **Resource** > **Python**.
   ii. In the **Create Resource** dialog box, set **Resource Name** and **Location**.
   iii. Click **OK**.
   iv. On the configuration tab that appears, edit the code of the created resource. Sample code:

```
from odps.udf import annotate
@annotate("string->bigint")
class ipint(object):
    def evaluate(self, ip):
        try:
            return reduce(lambda x, y: (x << 8) + y, map(int, ip.split('.')))
        except:
            return 0
```

   v. Click in the toolbar.
2. Register a UDF.
   i. On the **Data Analytics** tab, move the pointer over and choose **MaxCompute** > **Function**. Alternatively, you can click a workflow in the Business Flow section, right-click **MaxCompute**, and then choose **Create** > **Function**.
   ii. In the **Create Function** dialog box, set **Function Name** and **Location**.
   iii. Click **Commit**.
   iv. In the **Register Function** section of the configuration tab that appears, enter the class name and the name of the Python resource that has been created, and then click in the toolbar. In this example, the class name is `ipint.ipint`.
   v. Check whether the ipint function is valid and meets your expectation. For example, you can create an ODPS SQL node to test the ipint function by running an SQL statement.

Reference and download resources

- For more information about how to reference resources for functions, see Register a UDF.
- For more information about how to reference resources for nodes, see Create an ODPS MR node.

To download a resource, double-click **Resource** under the target workflow. In the resource list that appears, move the pointer over the required resource and click **Download**.

2.5.5.2.9. Register a UDF

DataWorks allows you to develop UDFs in Python and Java. This topic describes how to register a UDF.

Prerequisites

Before you register a UDF, you must upload the related resource.

Procedure

1. Log on to the DataWorks console.
2. Create a workflow. For more information, see Create a workflow.
3. Write Java code in the offline Java environment, compress the code to a JAR package, and upload the package as a JAR resource to DataWorks. For more information, see Create a JAR resource.
4. Create a UDF.
   i. Find the target workflow, right-click **MaxCompute**, and then choose **Create** > **Function**.
   ii. In the **Create Function** dialog box, set **Function Name** and **Location** and click **OK**.

iii. In the **Register Function** section of the configuration tab that appears, set the parameters.

Parameter	Description
Function Type	The type of the function. Valid values: **Mathematical Function**, **Aggregate Function**, **String Function**, **Date Function**, **Analytic Function**, and **Other**.
Engine Instance MaxCompute	The MaxCompute engine instance bound to the current workspace. By default, you cannot change the engine instance.
Function Name	The name of the function, which is used to reference the function in SQL. The function name must be globally unique and cannot be modified after the function is registered.
Owner	The owner of the function. By default, this parameter is automatically set.
Class Name	Required. The name of the class for implementing the function. **Note** If the resource type is Python, enter the class name in the Python resource name.Class name format. Do not include the .py extension in the resource name.
Resources	Required. The list of resources. You can search for existing resources in the current workspace in fuzzy match mode.
Description	The description of the function.
Expression Syntax	The instructions on how to use the function, for example, `test`.
Parameter Description	The description of supported input and output parameter types.
Return Value	Optional. The value to return. Example: 1.
Example	Optional. An example of the function.

5. Click in the toolbar.
6. Commit the function.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

2.5.5.3. EMR

2.5.5.3.1. Create an EMR MR node

You can create an EMR MR node to compute a large-scale dataset by using multiple Map tasks in a parallel manner.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **EMR > EMR MR**. Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR MR**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.3.2. Create an EMR Spark SQL node

You can create an EMR Spark SQL node to use the distributed SQL query engine to process structured data, improving the task execution efficiency.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **EMR > EMR Spark SQL**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Spark SQL**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.3.3. Create an EMR Spark node

You can create an EMR Spark node to perform complex memory analysis and build large and low-latency data analysis applications.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **EMR > EMR Spark**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Spark**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.

5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.3.4. Create an EMR Hive node

This topic describes how to create an EMR Hive node. This type of node allows you to use SQL-like statements to read data from, write data to, and manage data warehouses with a large amount of data stored in a distributed storage system. By using this type of node, you can efficiently analyze a large amount of log data.

Prerequisites

The EMR folder is available on the DataStudio page only after you bind an E-MapReduce compute engine to the current workspace on the **Project Management** page.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **EMR > EMR Hive**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **EMR**, and then choose **Create > EMR Hive**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. On the node configuration tab, select an E-MapReduce compute engine from the **Engine Instance EMR** drop-down list and edit the code of the node.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.4. Algorithm

2.5.5.4.1. Create a PAI node

PAI nodes are used to call tasks that are created on PAI and schedule production activities based on the node configuration.

Prerequisites

To create a PAI node in DataWorks, you must first create a PAI experiment in PAI.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **Machine Learning** > **PAI Experiment**. Alternatively, you can click a workflow in the Business Flow section, right-click **Algorithm**, and then choose **Create** > **PAI Experiment**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Select the PAI experiment that you have created from the **Experiment** drop-down list and load it.If you want to modify the PAI experiment, click **Edit in PAI Console**.
6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.5. General

2.5.5.5.1. Create a for-each node

This topic describes how to use a for-each node to repeat a loop twice and display the loop count.

Prerequisites

The MaxCompute module is available on the DataStudio page only after you bind a MaxCompute compute engine to the current workspace on the **Project Management** page.

Context

You can use a for-each node to repeat a loop for a maximum of 128 times. If the loop count exceeds this limit, an error occurs.

If the for-each node needs to perform logic judgment and result traversal, you can use the branch node. However, the branch node must be used with the merge node for result traversal.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > for-each**.

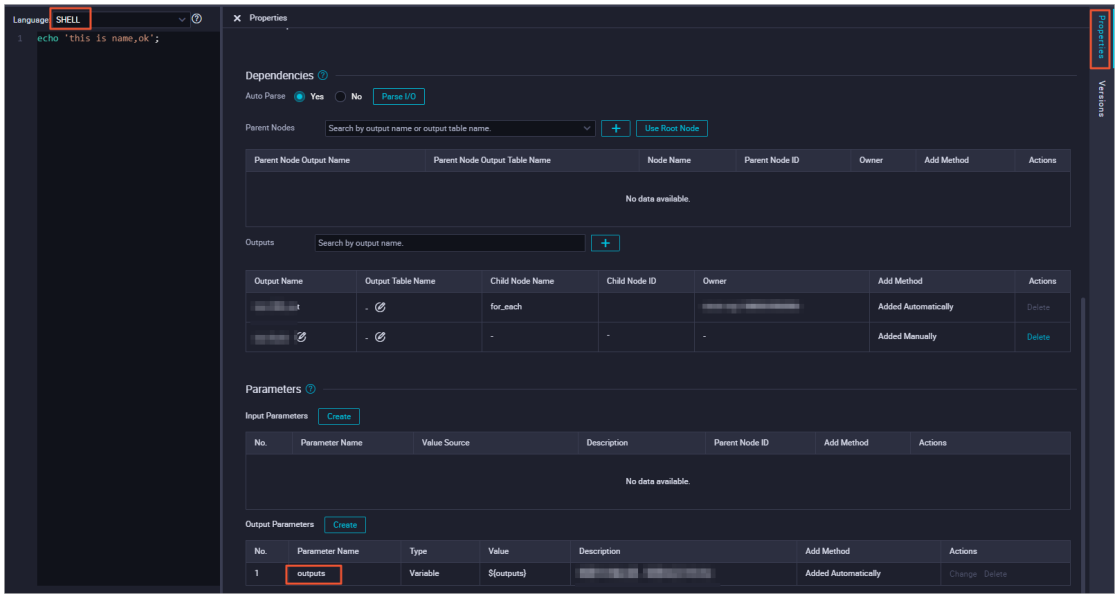
   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > for-each**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. Create a workflow with an assignment node as the parent node and a for-each node as the child node. For more information, see Create a workflow.

i. Double-click the created assignment node. Set the language of the assignment node to SHELL, and enter the following code:

```
echo 'this is name,ok';
```

On the node configuration tab, click the **Properties** tab in the right-side navigation pane. By default, the **outputs** parameter appears in the **Output Parameters** section.



ii. Double-click the created for-each node. Enter the following code for the for-each node:

```
echo ${dag.loopTimes} ----Display the loop count.
```

> **Note**
>
> - The start and end nodes of the for-each node have fixed logic and cannot be edited.
> - After you modify the code of the Shell node, save the modification. No message will appear to remind you to save the modification when you commit the node. If you do not save the modification, the code cannot be updated to the latest version in time.

A for-each node supports the following environment variables:

- ${dag.foreach.current}: the current data row.
- ${dag.loopDataArray}: the input dataset.
- ${dag.offset}: the offset of the loop count to 1.
- ${dag.loopTimes}: the loop count, whose value equals to the value of ${dag.offset} plus 1.

```
// Compare the code of the Shell node with that of a common for loop.
data=[]  // It is equivalent to ${dag.loopDataArray}.
// i is equivalent to ${dag.offset}.
for(int i=0;i<data.length;i++) {
  print(data[i]);  // data[i] is equivalent to ${dag.foreach.current}.
}
```

The ${dag.loopDataArray} parameter is the default input parameter of the for-each node. Set this parameter to the value of the outputs parameter of the parent node. If you do not set this parameter, an error occurs when you commit the node.

6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.

> **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click  in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the batch sync node. For more information, see Publish nodes.

8. Test the batch sync node. For more information, see Recurring tasks.

2.5.5.5.2. Create a do-while node

You can define mutually dependent nodes, including a loop decision node named end, in a do-while node. DataWorks repeatedly runs the nodes and exits the loop only when the end node returns False.

Context

> **Note** A loop can be repeated for a maximum of 128 times. If the loop count exceeds this limit, an error occurs.

The do-while node supports the MaxCompute SQL, SHELL, and Python languages. If you use MaxCompute SQL, you can use a `CASE WHEN` statement to evaluate whether the specified condition for exiting the loop is met.

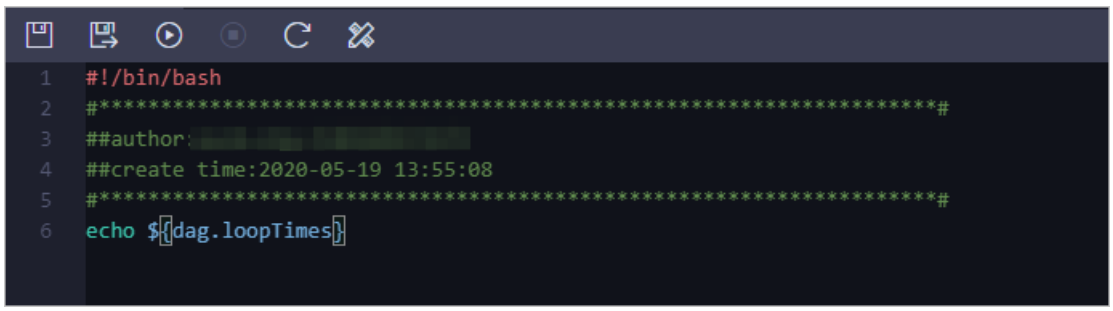
Simple example

This section describes how to use a do-while node to repeat a loop five times and display the loop count each time the loop runs.

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > do-while**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > do-while**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. Define the loop body.By default, the do-while node consists of the start, SQL, and end nodes.
   - The start node marks the startup of a loop and does not have any business effect.
   - DataWorks provides the SQL node as a sample business processing node. You must replace the SQL node with your own business processing node, for example, a Shell node named Display loop count.

   - The end node marks the end of a loop and determines whether to start the loop again. In this example, it defines the condition for exiting the loop for the do-while node.

The end node is an assignment node. It generates only True or False, indicating whether to start the loop again or exit the loop.

The ${dag.loopTimes} variable is used in both the Display loop count node and the end node. It is a reserved variable of DataWorks. This variable indicates the loop count and the value increments from 1. All internal nodes of the do-while node can reference this variable.

In the code shown in the preceding figure, the value of the `dag.loopTimes` variable is compared with 5 to limit the loop count. The value of the dag.loopTimes variable is 1 when the loop runs for the first time and is incremented by 1 each time, for example, 2 for the second time. In the fifth loop, the value is 5. In this case, the result of ${dag.loopTimes}<5 is False, and the do-while node exits the loop.

6. Run the do-while node. You can configure the scheduling properties for the do-while node as needed and commit it to **Operation Center** for running.
   - do-while node: The do-while node appears as a whole node in Operation Center. To view the loop details about the do-while node, right-click the node in the DAG and select **View Internal Nodes**.
   - Internal loop body: This view is divided into three parts.
     - The left pane of the view lists the rerun history of the do-while node. A record is generated each time a do-while node instance is run.
     - The middle pane of the view shows a loop record list. A record is generated each time the loop of the do-while node is run. The running status of each loop also appears.
     - The right pane of the view shows the details about the do-while node each time the loop is run. You can click a record in the loop record list to view the running details.
7. View the running result. View the internal loop body. In the loop record list, click the record corresponding to the third loop. The loop count is 3 in the runtime logs.

   You can also view the runtime logs of the end node that are generated when the loop runs for the third time and for the fifth time, respectively.

   Based on the preceding simple example, the do-while node works in the following way:

   i. Run from the start node.
   ii. Run nodes in sequence based on the defined node dependencies.
   iii. Define the condition for exiting the loop in the end node.
   iv. Run the conditional statement of the end node after the loop ends for the first time.
   v. Record the loop count as 1 and start the loop again if the conditional statement returns True in the runtime logs of the end node.
   vi. Exit the loop if the conditional statement returns False in the runtime logs of the end node.

Complex example

In addition to simple scenarios, do-while nodes can also be used in complex scenarios where each row of data is processed in sequence by using a loop. Before you process data in such scenarios, make sure that:

- You have deployed a parent node that can export queried data to the do-while node. You can use an assignment node to meet this condition.

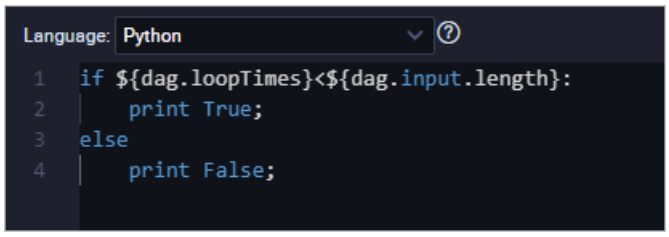
- The do-while node can obtain the output of the parent node. You can configure the node context and dependencies to meet this condition.
- The internal nodes of the do-while node can reference each row of data. In this example, the existing node context is enhanced and the system variable ${dag.offset} is used to reference the context of the do-while node.

This section describes how to use the do-while node to display the data entries in a table in sequence until all data entries in the table are displayed. Each time the loop runs, a data entry is displayed.

1. On the **Data Analytics** tab, double-click the created do-while node.
2. Define the loop body.
   i. Create an assignment node named Initialize dataset and add it as the parent node of the do-while node. The parent node generates a test dataset.
   ii. On the Properties tab of the do-while node, define an input parameter in the **Parameters** section. Set Parameter Name to input and Value Source to the output of the parent node.
   iii. Write code for the business processing node named Print each data row.
      - `${dag.offset}` : a reserved variable of DataWorks. This variable indicates the offset of the loop count to 1. For example, the offset is 0 when the loop runs for the first time and 1 for the second time. The offset equals to the loop count minus 1.
      - `${dag.input}` : the context that you configure for the do-while node. In the preceding steps, the input parameter is defined for the do-while node and the value of the input parameter is the output of the parent node named Initialize dataset.

        The internal nodes of the do-while node can directly use ${dag.${ctxKey}} to reference the context. In this example, ${ctxKey} is set to input. Therefore, you can use ${dag.input} to reference the context.
      - `${dag.input[${dag.offset}]}` : the data obtained from the table generated by the Initialize dataset node. DataWorks can obtain a row of data from the table based on the specified offset. The value of the ${dag.offset} variable increments from 0. Therefore, the data entries such as ${dag.input[0]} and ${dag.input[1]} are returned until all data entries in the dataset are returned.

iv. Define the condition for exiting the loop for the end node. The values of the ${dag.loopTimes} and ${dag.input.length} variables are compared, as shown in the following figure. If the value of the former is less than that of the latter, the end node returns True and the do-while node continues the loop. Otherwise, the end node returns False and the do-while node exits the loop.



Note The system automatically sets the ${dag.input.length} variable to the number of rows in the array specified by the input parameter based on the context configured for the do-while node.

3. Run the do-while node and view the running result.

Summary

- Compared with the while, foreach, and do...while statements, a do-while node has the following characteristics:
  - A do-while node contains a loop body that runs a loop before evaluating the conditional statement. This node functions the same as the do...while statement. A do-while node can also use the system variable ${dag.offset} and the node context to implement the feature of the foreach statement.
  - A do-while node cannot achieve the feature of the while statement because a do-while node runs a loop before evaluating the conditional statement.
- A do-while node works in the following way:
  i. Run nodes in the loop body starting from the start node based on node dependencies.
  ii. Run the code defined for the end node.
    - Run the loop again if the end node returns True.
    - Exit the loop if the end node returns False.
- How to use the node context: The internal nodes of a do-while node can use ${dag.${ctxKey}} to reference the context defined for the do-while node.
- System parameters: DataWorks provides the following system variables for the internal nodes of the do-while node:
  - ${dag.loopTimes}: the loop count, starting from 1.
  - ${dag.offset}: the offset of the loop count to 1, starting from 0.

2.5.5.5.3. Create an OSS Object Inspection node

If descendant nodes depend on specified OSS objects, you can use the OSS object inspection feature to check whether the OSS objects exist. For example, you can run a node for synchronizing OSS data files to DataWorks only after the OSS data files are generated. In this case, you can use the OSS object inspection feature to monitor the OSS data files.

The OSS object inspection feature can monitor OSS objects of all tenants. To create an OSS Object Inspection node, perform the following steps:

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General** > **OSS Object Inspection**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create** > **OSS Object Inspection**.
3. In the **Create Node** dialog box, set **Node Name** and **Location** and click **Commit**.

   > **Note** The node name must be 1 to 128 characters in length.
4. In the **OSS Object Inspection** section of the node configuration tab that appears, set the parameters.

No.	Parameter	Description
1	**OSS Object**	The storage path of the OSS object. You can add a scheduling parameter to the storage path.
2	**Timeout**	The timeout period during which DataWorks checks whether the OSS object exists in OSS every 5 seconds. If the OSS object is not detected before the timeout period ends, the OSS Object Inspection node fails.
3	**Storage Address**	The storage space of the OSS object. Valid values: ○ **Myself**: detects the OSS object in the storage space of the current tenant. ○ **Other**: detects the OSS object in the storage space of another tenant.

   > **Note**
   > ○ When an OSS Object Inspection node is running, it monitors the OSS object by using MaxCompute. Make sure that MaxCompute has the required permissions on the OSS bucket.
   > ○ In the development or production environment, the node monitors the OSS object by using the access identity of the development or production environment. Make sure that the access identity has the required permissions on the OSS bucket.
   > ○ You cannot specify an OSS object by using the wildcard (*), nor can you use the system parameters cyctime and bizdate. However, you can use custom parameters.
5. Grant MaxCompute the permission to access OSS in the RAM console.

   MaxCompute uses RAM and Security Token Service (STS) of Alibaba Cloud to resolve security issues of accounts.

   ○ If the owners of the MaxCompute project and OSS bucket are using the same Apsara Stack

tenant account, you can authorize MaxCompute to access OSS with one click in the RAM console.

- If the owners of the MaxCompute project and OSS bucket are using different Apsara Stack tenant accounts, you can perform the following steps to authorize MaxCompute to access OSS:

  a. Create a role in the RAM console.

  Create a role, such as AliyunODPSDefaultRole or AliyunODPSRoleForOtherUser, and set the following policy:

```
--The owners of the MaxCompute project and OSS bucket are using different Apsara Stack tenant accounts.
{
"Statement": [
{
"Action": "sts:AssumeRole",
"Effect": "Allow",
"Principal": {
"Service": [
"ID of the Apsara Stack tenant account that owns the MaxCompute project@odps.aliyuncs.com"
]
}
}
],
"Version": "1"
}
```

b. Create the AliyunODPSRolePolicy permission policy that contains the permissions required for accessing OSS.

```
{
"Version": "1",
"Statement": [
{
"Action": [
 "oss:ListBuckets",
 "oss:GetObject",
 "oss:ListObjects",
 "oss:PutObject",
 "oss:DeleteObject",
 "oss:AbortMultipartUpload",
 "oss:ListParts"
],
"Resource": "*",
"Effect": "Allow"
}
]
}
--You can also add other permissions as required.
```

c. Grant the AliyunODPSRolePolicy permission policy to the role.

6. Go to **Operation Center** to view the runtime logs.

   If the following error information appears, the OSS object is not detected:

```
<Error>
 <Code>NoSuchKey</Code>
 <Message>The specified key does not exist. </Message>
 <RequestId></RequestId>
 <HostId>OSS object</HostId>
 <Key>xc/111.txt</Key>
</Error>
```

2.5.5.5.4. Create a merge node

This topic describes the definition of merge nodes and how to create a merge node and define the merging logic. It also provides an example to show the scheduling configuration and running details of a merge node.

A merge node is a logical control node in DataStudio. It can merge the running results of its parent nodes, regardless of their running statuses. It aims at facilitating the running of nodes that depend on the output of the child nodes of a branch node.

You cannot change the running status of a merge node. A merge node merges the running results of multiple child nodes of a branch node and sets the running status to Successful. To guarantee the proper running of a node that depends on the output of the child nodes of a branch node, you can configure the node to directly depend on the merge node.

For example, Branch node C has two logically exclusive branches C1 and C2. These two branches use different logic to write data to the same MaxCompute table. Assume that Node B depends on the output of this MaxCompute table. To make sure that Node B can run properly, you must use Merge node J to merge the running results of branches C1 and C2, and then configure Merge node J as the parent node of Node B. If Node B directly depends on branches C1 and C2, one of the branches will fail to run because only one branch meets the branch condition each time Branch node C runs. In this case, Node B cannot be triggered as scheduled.

Create a merge node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over  and choose **General > MERGE Nodes**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > MERGE Nodes**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.

Define the merging logic

After the merge node is created, the node configuration tab appears. Specify the branches to be merged for the node. Enter the output name or output table name of the parent node, and click the **Add** icon. You can view the running status in the Result section. The available running statuses are **Successful** and **Branch Not Running**.

Click the **Properties** tab in the right-side navigation pane and configure the scheduling properties of the merge node.

Run the merge node

If a branch meets the specified condition, the branch is run. You can select the branch and view the running details on the **Runtime Logs** tab.

If a branch does not meet the specified condition, the branch is skipped. You can select the branch and view related information on the **Runtime Logs** tab.

2.5.5.5.5. Create a branch node

A branch node is a logical control node in DataStudio. It can define the branch logic and the direction of branches under different logical conditions.

Prerequisites

Generally, branch nodes need to be used with assignment nodes.

Create a branch node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > Branch Node**.Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Branch Node**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Define the branch logic
   i. In the **Definition** section, click **Add Branch**.
   ii. In the **Branch Definition** dialog box, set the parameters.

Parameter	Description
Condition	The condition of the branch. ■ You can only use Python comparison operators to define logical conditions for the branch node. ■ If the result of the expression is *true* when the node is running, the corresponding branch condition is met. ■ If the expression fails to be parsed when the node is running, the whole branch node fails. ■ To define branch conditions, you can use global variables and parameters defined in the node context. For example, the ${input} variable can be used as an input parameter of the branch node.
Associated Node Output	The associated node output of the branch. ■ The node output is used to configure dependencies for the child nodes of the branch node. ■ If the branch condition is met, the child node corresponding to the node output is run. If the child node also depends on the output of other nodes, the status of these nodes is considered. ■ If the branch condition is not met, the child node corresponding to the node output is not run. The child node is set to the `Not Running` state.
Description	The description of the branch. For example, the branches ${input}==1 and ${input}>2 are defined.

iii. Click **OK**.After you add a branch, you can click **Change** or **Delete** in the Actions column of the branch to modify or delete it.

- Click **Change** to modify the branch and related dependencies.
- Click **Delete** to delete the branch and related dependencies.

6. On the configuration tab of the branch node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section.After the branch conditions are defined, the output names are automatically added to the **Outputs** section on the **Properties** tab. Then, you can associate child nodes with the branch node based on the output names.

> **Note**
>
> - Child nodes inherit dry-run properties of the parent node. Therefore, we recommend that you do not create a node depending on its last-cycle instance as the branch.
> - The dependencies established by drawing lines between nodes on the dashboard of a workflow are not recorded on the Properties tab. You must manually enter these dependencies.

7. Commit the node.

> **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click  in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.

8. Test the node.

Supported Python comparison operators

In the following table, assume that the value of the a variable is 10 and that of the b variable is 20.

Comparison operator	Description	Example
==	Equal: checks whether two objects are equal.	(a==b) returns false.
! =	Not equal: checks whether two objects are not equal.	(a! =b) returns true.
<>	Not equal: checks whether two objects are not equal.	(a<>b) returns true. This operator is similar to ! =.

Comparison operator	Description	Example
>	Greater than: checks whether the variable on the left side of the operator is greater than that on the right side.	(a>b) returns false.
<	Less than: checks whether the variable on the left side of the operator is less than that on the right side. If the return result is 0 or 1, 0 indicates false and 1 indicates true. These two results are equivalent to the special variables true and false, respectively.	(a<b) returns true.
>=	Greater than or equal to: checks whether the variable on the left side of the operator is greater than or equal to that on the right side.	(a>=b) returns false.
<=	Less than or equal to: checks whether the variable on the left side of the operator is less than or equal to that on the right side.	(a<=b) returns true.

2.5.5.5.6. Create an assignment node

An assignment node uses one of the three value assignment languages MaxCompute SQL, SHELL, and Python to assign values by using the outputs parameter. This node is used to transmit data between a parent node and a child node based on context-based parameters.

Context

The outputs parameter has the following limits:

- The value of the outputs parameter is taken only from the output of the last line of the code.
  - If you use MaxCompute SQL, the output of the SELECT statement in the last line is used.
  - If you use SHELL, the output of the ECHO statement in the last line is used.
  - If you use Python, the output of the PRINT statement in the last line is used.
- The passed value of the outputs parameter is limited to 2 MB in size. If the output of the assignment statement exceeds this limit, the assignment node fails to run.

> **Note** If you use Python or SHELL, the value of the outputs parameter is a one-dimensional array where elements are separated with commas (,). If you use MaxCompute SQL, the value of the outputs parameter is passed to child nodes as a two-dimensional array.

Create an assignment node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > Assignment Node**.

Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Assignment Node**.

3. In the **Create Node** dialog box, set **Node Name** and **Location**.

> **Notice** A node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

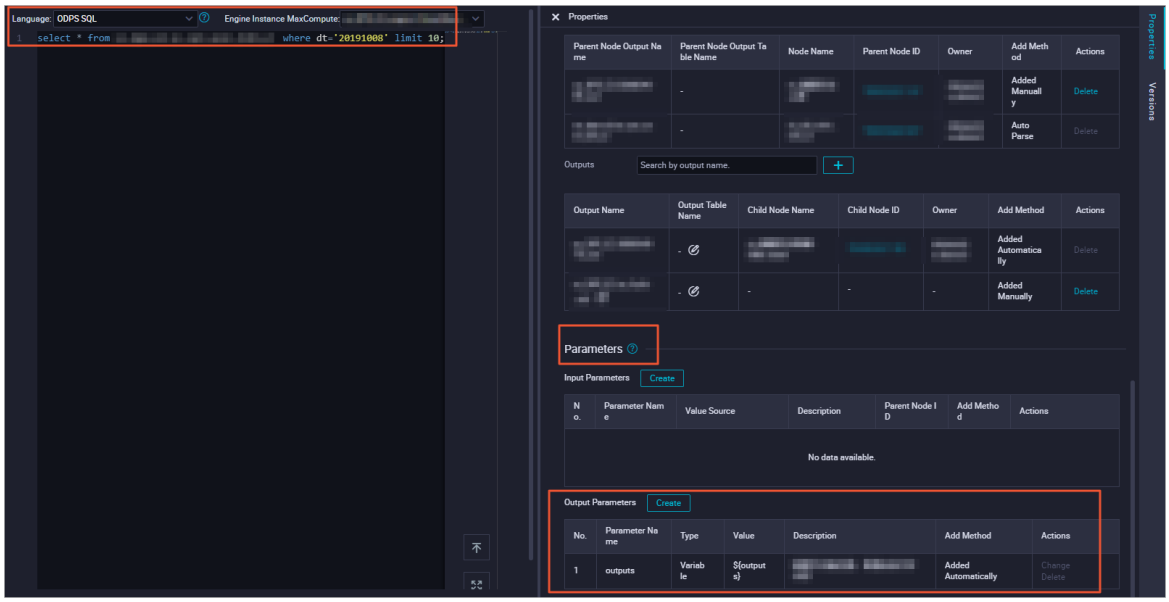
This following sections describe how to use assignment nodes that use the Python, MaxCompute SQL, and SHELL languages respectively to pass data between a parent node and a child node named Assignment node value comparison_shell by using context-based parameters.

After the assignment nodes that use the Python, MaxCompute SQL, and SHELL languages are created, you must set the dependencies so that the child node can reference the parameter values passed by these nodes.

4. Click **Commit**.

Configure the child node to reference the output values of the assignment node that uses MaxCompute SQL

1. Find the target workflow and double-click the assignment node fuzhi_sql that uses MaxCompute SQL.
2. On the configuration tab of the fuzhi_sql node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_sql node.The fuzhi_sql node assigns the results queried from a specified table to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_sql node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.The Assignment node value comparison_shell node depends on the fuzhi_sql node and uses the value of the outputs parameter of the fuzhi_sql node as the value of its input parameter

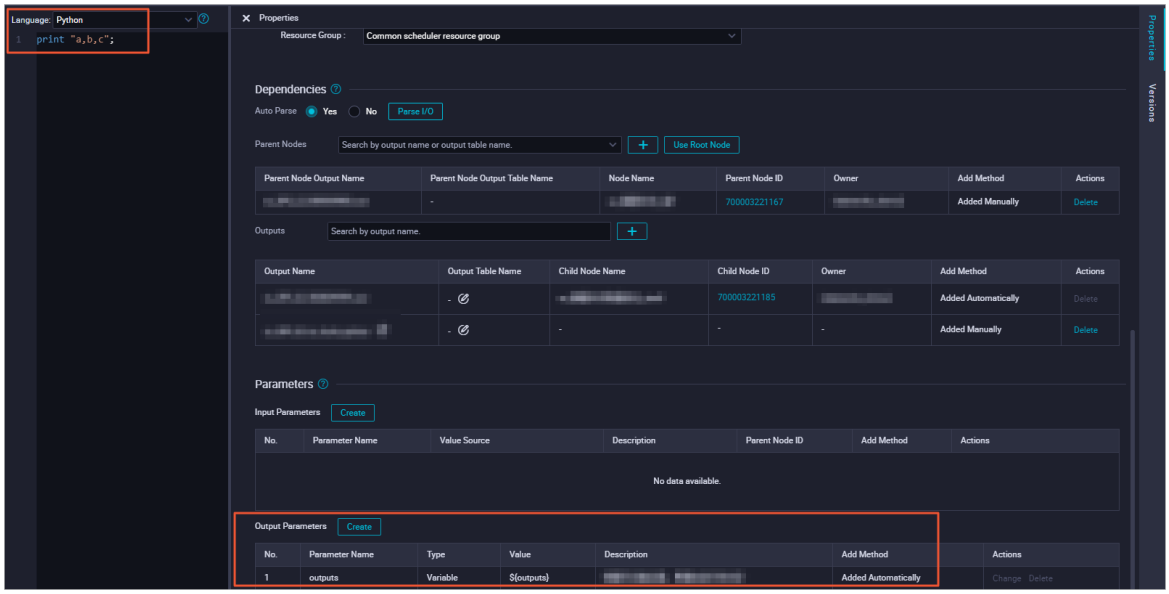
sql_inputs.

```
echo '${sql_inputs}';
echo 'Use the value in the first line in the output of the fuzhi_sql node as the input'${sql_inputs[0]};
echo 'Use the value in the second line in the output of the fuzhi_sql node as the input'${sql_inputs[1]};
echo 'Use the value of the second field in the first line in the output of the fuzhi_sql node as the input'${sql_inputs[0][1]};
echo 'Use the value of the third field in the second line in the output of the fuzhi_sql node as the input'${sql_inputs[1][2]};
```

6. Click in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

Configure the child node to reference the output values of the assignment node that uses Python

1. Find the target workflow and double-click the assignment node fuzhi_python that uses Python.
2. On the configuration tab of the fuzhi_python node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_python node.The fuzhi_python node assigns the values a,b,c to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_python node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.The Assignment node value comparison_shell node depends on the fuzhi_python node and uses

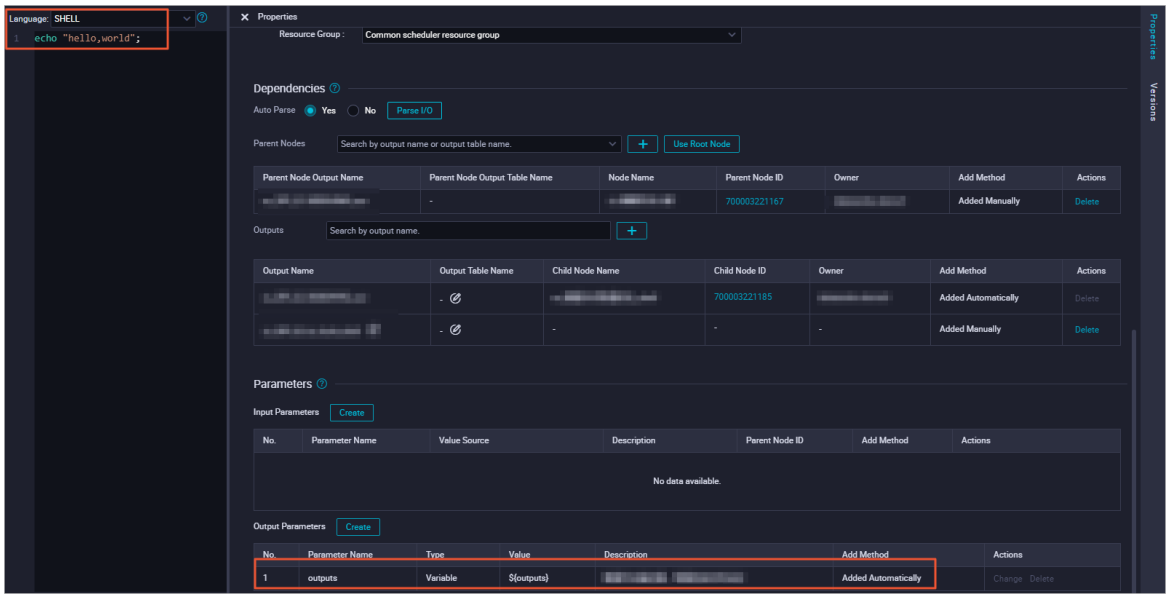
the value of the outputs parameter of the fuzhi_python node as the value of its input parameter python_inputs.

```
echo 'The output of the fuzhi_python node'${python_inputs};
echo 'Use the first value in the output of the fuzhi_python node as the input'${python_inputs[0]};
echo 'Use the second value in the output of the fuzhi_python node as the input'${python_inputs[1]};[1]}
```

6. Click in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

Configure the child node to reference the output values of the assignment node that uses SHELL

1. Find the target workflow and double-click the assignment node fuzhi_shell that uses SHELL.
2. On the configuration tab of the fuzhi_shell node that appears, click **Properties** in the right-side navigation pane.
3. Configure the fuzhi_shell node.The fuzhi_shell node assigns the values hello,world to the outputs parameter.



4. Double-click the Assignment node value comparison_shell node, which is the child node of the fuzhi_shell node.
5. On the configuration tab of the Assignment node value comparison_shell node that appears, click **Properties** in the right-side navigation pane and configure the node.The Assignment node value comparison_shell node depends on the fuzhi_shell node and uses the value of the outputs parameter of the fuzhi_shell node as the value of its input parameter shell_inputs.

```
echo 'The output of the fuzhi_shell node'${shell_inputs};
echo 'Use the first value in the output of the fuzhi_shell node as the input'${shell_inputs[0]};
echo 'Use the second value in the output of the fuzhi_shell node as the input'${shell_inputs[1]};
```

6. Click in the toolbar.
7. In the **Warning** message, click **Continue to Run**.
8. View the result.

2.5.5.5.7. Create a Shell node

Shell nodes support standard shell syntax but not interactive syntax.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General** > **Shell**.Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create** > **Shell**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. Edit the Shell node.
   i. Edit the code on the configuration tab of the Shell node.To call the system scheduling parameters for the Shell node, execute the following statement:

   ```
   echo "$1 $2 $3"
   ```

   > **Note** Separate multiple parameters with spaces.

   i. Click in the toolbar to save the SQL statement to the server.

   ii. Click in the toolbar to execute the SQL statement you have saved.If you need to change the resource group used to test the Shell node on the **DataStudio** page, click in the toolbar and select your desired exclusive resource group.
6. On the configuration tab of the Shell node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section.
7. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

i. Click in the toolbar.

ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

iii. Click **OK**.

In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.

8. Test the node.

2.5.5.5.8. Create a zero-load node

A zero-load node is a control node, which only supports dry-run scheduling and does not generate any data. It usually serves as the root node of a workflow.

Context

You can configure an output table for a zero-load node so that the output table can be used as an input table of another node. However, the zero-load node does not process the table data.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, move the pointer over and choose **General > Zero-Load Node**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Zero-Load Node**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the configuration tab of the zero-load node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
6. Commit the node.

   > **Notice** You must set **Rerun** and **Parent Nodes** before you can commit the node.

   i. Click in the toolbar.

   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.

   iii. Click **OK**.

   In a workspace in standard mode, you need to click **Deploy** in the upper-right corner after you commit the branch node.
7. Test the node.

2.5.5.5.9. Create a cross-tenant collaboration node

Cross-tenant collaboration nodes are used to associate nodes from different tenants. Cross-tenant collaboration nodes are classified into sender nodes and receiver nodes.

Prerequisites

A sender node and its receiver node use the same CRON expression. You can click the **Properties** tab in the right-side navigation pane of a node configuration tab and view the CRON expression in the **Schedule** section.

Create a cross-tenant collaboration node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > Cross-Tenant Collaboration**.

   Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Cross-Tenant Collaboration**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. On the node configuration tab, set the parameters in the **Cross-Tenant Collaboration** section.

Parameter	Description
Type	The type of the cross-tenant collaboration node. Valid values: **Sender** and **Receiver**.
Location	The path of the cross-tenant collaboration node. The node path cannot be modified.
Collaborative Workspaces	The workspace name and Apsara Stack tenant account of the peer node. This example sets the node type to Sender. Therefore, you must enter the workspace name and Apsara Stack tenant account of the receiver node.

5. After the sender node is created, follow the same procedure to create the receiver node under the Apsara Stack tenant account and workspace to which the receiver node belongs.

   Set the node type to **Receiver**. The information about available sender nodes appears. You must also set **Timeout**. This parameter indicates the timeout period of the receiver node after it starts running.

   The sender node first sends a message to the message center. After the message is delivered, the status of the sender node is set to successful. The receiver node continuously pulls messages from the message center. If a message is received within the timeout period, the status of the receiver node is set to successful.

   If the receiver node does not receive any messages within the timeout period, the receiver node fails. The lifecycle of a message is 24 hours.

Assume that an auto triggered instance was run on October 8, 2018. A message indicating the completion of the instance was then sent to the message center. If you create a retroactive instance for the receiver node with the data timestamp set to October 7, 2018, the status of the generated receiver node instance is set to successful.

6. After the configuration is completed, save and commit the node.

2.5.5.5.10. Create a data analysis report node

A data analysis report node is used to associate a report in the DataAnalysis module with the parent nodes on which the report depends and update the report as scheduled.

Prerequisites

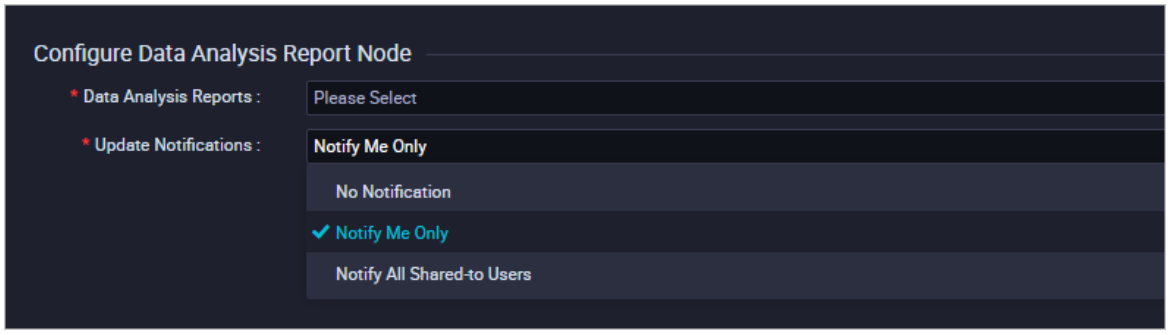
A table is created on the **Report** page of the **DataAnalysis** module.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > Data Analysis Reports**. Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Data Analysis Reports**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   Note The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the node configuration tab, set the parameters in the **Configure Data Analysis Report Node** section.



Parameter	Description
Data Analysis Reports	The report for which you want to receive the notifications about the updates. Select a report created on the **Report** page of the **DataAnalysis** module.
Update Notifications	Specifies the users who can receive the notifications when the report is updated. Valid values: **No Notification**, **Notify Me Only**, and **Notify All Shared to Users**.

6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side

navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.

7. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the data analysis report node. For more information, see Recurring tasks.After the node is run in the production environment, you can view updates of the report on the Report page of the **DataAnalysis** module.

2.5.5.5.11. Create a real-time sync node check node

This topic describes how to create and configure a real-time sync node check node.

Context

DataWorks provides the real-time sync node check node for you to associate the status of the current node with scheduling of its child node, which is a batch sync node. After you set dependencies in the workflow, the scheduling of the child node is determined by the status of the current node. This node allows you to schedule real-time sync nodes and batch sync nodes in a unified manner.

Create a real-time sync node check node

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **General > Check Realtime Compute Node**. Alternatively, you can click a workflow in the Business Flow section, right-click **General**, and then choose **Create > Check Realtime Compute Node**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   Note The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
4. Click **Commit**.
5. On the node configuration tab that appears, set the parameters in the **Check Realtime Compute Node** section.

Parameter	Description
StreamCompute Node	The real-time sync node in Stream Studio. Note The node must be deployed in Stream Studio.

Parameter	Description
Control rule	The control rule that determines whether to schedule or block the child node based on the status of the current node. ○ The status of the current node can be **Normal** or **Paused**. ■ **Normal**: The node is running and no error is reported. ■ **Paused**: The node is stopped or paused manually, or the node stops or cannot be restarted due to an exception. ○ The control method for the child node can be **Scheduled** or **Blocked**. ■ **Scheduled**: The child node can be run as scheduled. ■ **Blocked**: The child node cannot be scheduled, regardless of the settings.

6. Commit the node.
   i. Click in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the real-time sync node check node. For more information, see Publish nodes.

2.5.5.6. Custom

2.5.5.6.1. Create a Hologres development node

This topic describes how to create and modify a Hologres development node and update the node version.

Procedure

1. Log on to the DataWorks console.
2. On the Data Analytics tab, move the pointer over and choose **Custom > Hologres Development**. Alternatively, you can click a workflow in the Business Flow section, right-click **UserDefined**, and then choose **Create > Hologres Development**.
3. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

4. Click **Commit**.
5. On the node configuration tab that appears, select a Hologres development node.

If no Hologres node is available, click **Create** to create one. You can also click **Change** to modify an existing node.

6. On the configuration tab of the batch sync node, click the **Properties** tab in the right-side navigation pane. On the Properties tab, set parameters in the Schedule section. For more information, see Basic properties.
7. Commit the node.
   i. Click  in the toolbar.
   ii. In the **Commit Node** dialog box, enter your comments in the **Description** field.
   iii. Click **OK**.

   In a workspace in standard mode, you must click **Deploy** in the upper-right corner after you commit the node. For more information, see Publish nodes.
8. Test the batch sync node. For more information, see Recurring tasks.

2.5.6. Schedule

2.5.6.1. Basic properties

On the Properties tab of a node, you can set parameters of the node in the General, Schedule, Dependencies, and Parameters sections. The General section allows you to set the basic properties of the node.

On the **Data Analytics** tab of the DataStudio page, double-click a node. On the node configuration tab that appears, click the **Properties** tab in the right-side navigation pane and set the parameters in the **General** section.

Parameter	Description
Node Name	The name of the node that you set when creating the node. To modify the name, right-click the node in the left-side navigation pane and select **Rename**.
Node ID	The unique ID of the node. The node ID is generated when the node is committed at the first time. The node ID cannot be modified.
Node Type	The type of the node that you set when creating the node. The node type cannot be modified.
Owner	The owner of the node. By default, the owner of a newly created node is the current logon user. You can change the owner. **Note** Only a member in the workspace where the node resides can be selected as the owner.
Description	The description of the node, such as the business and usage.
Arguments	The parameter used to assign a value to a variable in the code during node scheduling. You can enter multiple parameters. Separate multiple parameters with spaces.

Parameter value assignment formats for various node types

- Format for ODPS SQL and ODPS MR nodes: `Variable name 1=Parameter 1 Variable name 2=Parameter 2`. Separate multiple parameters with spaces.
- Format for Shell nodes: `Parameter 1 Parameter 2`. Separate multiple parameters with spaces.

For more information about the built-in scheduling parameters, see Parameter configuration.

2.5.6.2. Scheduling parameters

In common data development scenarios, the code of different types of nodes may be subject to change from time to time. You must dynamically modify the values of some parameters, such as the date and time, based on the requirement changes and time changes.

In this case, you can use the scheduling parameter configuration feature of DataWorks. After relevant parameters are set, auto triggered nodes can automatically parse the code to obtain required data. Configurable parameters in DataWorks are classified into system parameters and custom parameters. We recommend that you use custom parameters.


```
{
  "data":[
  {
    "beginRunningTime":"1564019679966",
    "beginWaitResTime":"1564019679966",
    "beginWaitTimeTime":"1564019679506",
    "bizdate":"1559318400000",
    "createTime":"1564019679464",
    "dagId":332455685,
    "dagType":5,
    "finishTime":"1564019679966",
    "instanceId":2427622331,
    "modifyTime":"1564019679966",
    "nodeName":"vi","status":6
  }
  ],
  "errCode":"0",
  "errMsg":"",
  "requestId":"E17535-8C06-43F6-B1EA-6236FE9",
  "success":true
  }
```


Auto-completion is supported when you specify a data type for a parameter.

Parameter types

Parameter type	Configuration method	Applicable to	Example
System parameters: including bdp.system.bizdate and bdp.system.cyctime	To use the system parameters in the scheduling system, reference ${bdp.system.bizdate} and ${bdp.system.cyctime} in the code, instead of setting them in the Arguments field. The system can automatically replace the values of the parameters that reference the system parameters in the code.	All nodes	N/A
Non-system parameters: custom parameters (recommended)	Reference ${key1} and ${key2} in the code and set them in the Arguments field, for example, `"key1=value1 key2=value2"`.	Non-Shell nodes	• **Constant parameters:** param1="abc"param2=1234. • **Variables:** param1=$[yyyymmdd], the value of which is calculated based on the value of bdp.system.cyctime.
	Reference $1, $2, and $3 in the code and set them in the Arguments field, for example, `"value1 value2 value3"`.	Shell nodes	• **Constant parameters:** "abc" 1234. • **Variables:** $[yyyymmdd], the value of which is calculated based on the value of bdp.system.cyctime.

As described in the preceding table, the values of custom variables are calculated based on the values of system parameters. You can use custom variables to flexibly define the data to be obtained and the data format. For custom parameters, the following types of brackets are used:

- Braces { } define the data timestamp. For example, the value of {yyyymmdd} is calculated based on the value of bdp.system.bizdate.
- Brackets [ ] define the running time. For example, the value of [yyyymmddhh] is calculated based on the value of bdp.system.cyctime.

> **Note** Nodes can be scheduled only in the production environment. Therefore, the values of scheduling variables are replaced only after nodes are run in the production environment.
>
> After you set the scheduling variables for a node, you can click the **Run Smoke Test in Development Environment** icon on the node configuration tab to test whether the values of scheduling variables can be replaced as expected during node scheduling.
>
> You can click the **Properties** tab in the right-side navigation pane, and assign values to scheduling variables in the **Arguments** field in the **General** section. Note the following issues when you set parameters:
>
> - Do not add spaces on either side of the equal sign (=) for a parameter. For example, enter `bizdate=$bizdate`.
> - Separate multiple parameters (if any) with spaces. For example, enter `bizdate=$bizdate datetime=${yyyymmdd}`.

System parameters

DataWorks provides the following system parameters:

- ${bdp.system.cyctime}: the scheduled time to run an instance. Default format: yyyymmddhh24miss. This parameter can specify the hour and minutes of the scheduled time.
- ${bdp.system.bizdate}: the timestamp of data to be analyzed by an instance. Default format: yyyymmdd. The default data timestamp is one day before the scheduled time.

Use the following formula to calculate the running time based on the data timestamp: `Running time = Data timestamp + 1`.

To use the system parameters, you can reference them in the code, instead of setting them in the Arguments field. The system can automatically replace the values of the parameters that reference the system parameters in the code.

> **Note** The scheduling properties of an auto triggered node are configured to define the scheduling rules of the running time. Therefore, you can calculate the data timestamp based on the scheduled time to run an instance and obtain the values of system parameters for the instance.

Example of system parameters

For example, to set an ODPS SQL node to run once per hour from 00:00 to 23:59 every day, perform the following steps if you want to use system parameters in the code:

1. Reference system parameters in the code.

```
insert overwrite table tb1 partition(ds ='20150304') select
c1,c2,c3
from (
select * from tb2
where ds ='${bdp.system.cyctime}') t
full outer join(
select * from tb3
where ds = '${bdp.system.bizdate}') y
on t.c1 = y.c1;
```

2. After the preceding step, your node is partitioned by using the system parameters. Set the scheduling properties and dependencies. For more information, see Schedule and Dependencies. In this example, the node is scheduled by hour.
3. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Operation Center. The scheduling system generates instances for the auto triggered node from the second day. You can right-click an instance in the directed acyclic graph (DAG) and select **View Runtime Log** to view the parsed values of the system parameters.

   For example, the scheduling system generated 24 running instances for the node on January 14, 2019. The data timestamp is January 13, 2019 for all instances. Therefore, the value of ${bdp.system.bizdate} is 20190113. The running time is the running date appended with the scheduled time. Therefore, the value of ${bdp.system.cyctime} is 20190114000000 plus the scheduled time of each instance.

   Open the runtime logs of each instance and search for the replaced values of the system parameters in the code:

   - The scheduled time for the first instance is January 14, 2019 00:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114000000.
   - The scheduled time for the second instance is January 14, 2019 01:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114010000.
   - Similarly, the scheduled time for the twenty-fourth instance is January 14, 2019 23:00:00. Therefore, bdp.system.bizdate is replaced with 20190113 and bdp.system.cyctime is replaced with 20190114230000.

Custom parameters for non-Shell nodes

To set scheduling variables for a non-Shell node, add ${Variable name} in the code to reference the function and assign a value to the scheduling variable.

> **Note** The name of a variable in the SQL code can contain only letters, digits, and underscores (_). If the variable name is date, the value of $bizdate is automatically assigned to this variable. For more information, see the "Built-in scheduling parameters" section in this topic. You do not need to assign a value in the Arguments field. Even if another value is assigned, it is not used in the code because the value of $bizdate is automatically assigned in the code.

Example of custom parameters for non-Shell nodes

For example, to set an ODPS SQL node to run once per hour from 00:00 to 23:59 every day, perform the following steps if you want to use the hour-related custom variables thishour and lasthour in the code:

1. Reference the parameters in the code.

```
insert overwrite table tb1 partition(ds ='20150304')  select
 c1,c2,c3
from (
 select * from tb2
 where ds ='${thishour}') t
full outer join(
 select * from tb3
 where ds = '${lasthour}') y
on t.c1 = y.c1;
```

2. Click the **Properties** tab in the right-side navigation pane of the node configuration tab. Assign values to the custom parameters referenced in the code in the **Arguments** field in the **General** section.

   Set the custom parameters in the following formats:

   - thishour=$[yyyy-mm-dd/hh24:mi:ss]
   - lasthour=$[yyyy-mm-dd/hh24:mi:ss-1/24]

   > **Note** The value of yyyy-mm-dd/hh24:mi:ss corresponds to that of cyctime. For more information, see the "Custom parameters" section in this topic.

   You can enter `thishour=$[yyyy-mm-dd/hh24:mi:ss] lasthour=$[yyyy-mm-dd/hh24:mi:ss-1/24]` in the **Arguments** field.

3. Set the node to run once per hour.
4. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Operation Center. The scheduling system generates instances for the auto triggered node from the second day. You can right-click an instance in the DAG and select **View Runtime Log** to view the parsed values of the custom parameters. The value of cyctime is 20190114010000. Therefore, the value of thishour is 2019-01-14/01:00:00 and the value of lasthour, which indicates the last hour, is 2019-01-14/00:00:00.

Custom parameters for Shell nodes

The parameter configuration procedure of a Shell node is similar to that of a non-Shell node, except that the variable naming rules are different. Variable names for a Shell node cannot be customized, but must follow the $1,$2,$3... format. For example, add $1 in the code of a Shell node and enter the built-in scheduling parameter $xxx in the Arguments field. Then, the value of $xxx can replace that of $1 in the code.

> **Note** If the number of parameters in a Shell node reaches 10, use ${10} to declare the tenth variable.

Example of custom parameters for Shell nodes

For example, set a Shell node to run at 01:00 every day. To use the custom constant parameter myname and the custom variable ct in the code, perform the following steps:

1. Reference the parameters in the code.

   ```
   echo "hello $1, two days ago is $2, the system param is ${bdp.system.cyctime}";
   ```

2. Click the **Properties** tab in the right-side navigation pane of the node configuration tab. Assign values to the custom parameters referenced in the code in the **Arguments** field in the **General** section. Separate multiple parameters with spaces, for example, enter Parameter 1 Parameter 2 Parameter 3. The custom parameters are parsed based on the parameter sequence. For example, $1 is replaced with the value of Parameter 1. In this example, enter abcd $[yyyy-mm-dd-2] in the Arguments field to set $1 and $2 to abcd and $[yyyy-mm-dd-2], respectively.
3. Set the node to run at 01:00 every day.
4. After you set the recurrence and dependencies, commit and deploy the node. Then, you can check the node in Operation Center. The scheduling system generates instances for the auto triggered node from the second day. Right-click an instance in the DAG and select **View Runtime Log**. The logs show that $1 in the code is replaced with abcd, $2 is replaced with 2019-01-12 (two days before the running date), and ${bdp.system.cyctime} is replaced with 20190114010000.

Custom parameters

Custom parameters are divided into constant parameters and variables based on the value type. DataWorks provides some built-in scheduling parameters as variables.

- Constant parameters

  For example, for an SQL node, add ${Variable name} in the code and set the following parameter for the node: Variable name=Fixed value.

  - Code: `select xxxxxx type='${type}'`
  - Value assigned to the scheduling variable: type='aaa'. When the node is run, the variable in the code is replaced with type='aaa'.

- Variables

  Variables are built-in scheduling parameters whose values depend on the system parameters ${bdp.system.bizdate} and ${bdp.system.cyctime}.

  For example, for an SQL node, add ${Variable name} in the code and set the following parameter for the node: Variable name=Scheduling parameter.

  - Code: `select xxxxxx dt=${datetime}`
  - Value assigned to the scheduling variable: datetime=$bizdate

    If the node is run on July 22, 2017, the variable in the code is replaced with dt=20170721.

Built-in scheduling parameters

- $bizdate
  - Parameter description: the data timestamp in the format of yyyymmdd. By default, the value of this parameter is one day before the scheduled time to run a node.
  - For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizdate. If the node is run on July 22, 2017, $bizdate is replaced with pt=20170721.
- $cyctime
  - Parameter description: the scheduled time to run a node. If no scheduled time is configured for a node scheduled by day, $cyctime is set to 00:00 of the day. The time is accurate to seconds. This parameter is usually used for nodes scheduled by hour or minute.

    Note
    - Pay attention to the difference between the time parameters configured by using $[] and ${}. $bizdate specifies the data timestamp, which is one day before the current day by default.
    - $cyctime specifies the scheduled time to run a node. If no scheduled time is configured for a node scheduled by day, $cyctime is set to 00:00 of the day. The time is accurate to seconds. This parameter is usually used for nodes scheduled by hour or minute.

      For example, if a node is scheduled to run at 00:30 on the current day, $cyctime is set to yyyy-mm-dd 00:30:00.
    - If a time parameter is configured by using ${}, $bizdate is used as the benchmark for running nodes. The time parameter is replaced with the data timestamp selected for retroactive data generation.
    - If a time parameter is configured by using $[], $cyctime is used as the benchmark for running nodes. The time is calculated in the same way as the time in Oracle. The time parameter is replaced with the data timestamp selected for retroactive data generation plus one day.

      For example, if the data timestamp is set to 20140510 for retroactive data generation, $cyctime is replaced with 20140511.
  - The following examples show the values of custom parameters when $cyctime is set to 20140515103000:
    - $[yyyy] = 2014, $[yy] = 14, $[mm] = 05, $[dd] = 15, $[yyyy-mm-dd] = 2014-05-15, $[hh24:mi:ss] = 10:30:00, $[yyyy-mm-dd hh24:mi:ss] = 2014-05-1510:30:00
    - $[hh24:mi:ss - 1/24] = 09:30:00
    - $[yyyy-mm-dd hh24:mi:ss -1/24/60] = 2014-05-1510:29:00
    - $[yyyy-mm-dd hh24:mi:ss -1/24] = 2014-05-15 09:30:00
    - $[add_months(yyyymmdd,-1)] = 20140415
    - $[add_months(yyyymmdd,-12*1)] = 20130515
    - $[hh24] =10
    - $[mi] =30

  - Method for testing the $cyctime parameter:

    After an instance starts to run, right-click the instance in the DAG and select **More**. Check whether the scheduled time is the time at which the instance is run.

- $jobid
  - Parameter description: the ID of the workflow to which a node belongs.
  - Example: jobid=$jobid.
- $nodeid
  - Parameter description: the ID of a node.
  - Example: nodeid=$nodeid.
- $taskid
  - Parameter description: the instance ID of a node.
  - Example: taskid=$taskid.
- $bizmonth
  - Parameter description: the month of the data timestamp in the format of yyyymm. If the month of a data timestamp is the current month, the value of $bizmonth is the month of the data timestamp minus 1. Otherwise, the value of $bizmonth is the month of the data timestamp.
  - For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$bizmonth.

    Assume that the current day is July 22, 2017. If the node is run on July 22, 2017, $bizmonth is replaced with pt=201706.
- ${...} custom parameters
  - You can customize a time format based on the value of $bizdate, where yyyy indicates the four-digit year, yy indicates the two-digit year, mm indicates the month, and dd indicates the day. You can use any combination of these parameters, for example, ${yyyy}, ${yyyymm}, ${yyyymmdd}, and ${yyyy-mm-dd}.
  - $bizdate is accurate to the day. Therefore, ${...} can specify only the year, month, or day.

- The following table describes how to specify other intervals based on $bizdate.

Interval	Expression
N years later	${yyyy+N}
N years before	${yyyy-N}
N months later	${yyyymm+N}
N months before	${yyyymm-N}
N weeks later	${yyyymmdd+7*N}
N weeks before	${yyyymmdd-7*N}
N days later	${yyyymmdd+N}
N days before	${yyyymmdd-N}

- $gmtdate
  - Parameter description: the current date in the format of yyyymmdd. By default, the value of this parameter is the current date. During retroactive data generation, the input value is the data timestamp plus one day.
  - For example, the code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=$gmtdate. Assume that the current day is July 22, 2017. If the node is run on July 22, 2017, $gmtdate is replaced with pt=20170722.
- ${yyyymmdd}
  - Parameter description: the data timestamp in the format of yyyymmdd. The value of this parameter is the same as that of $bizdate. This parameter supports delimiters, for example, yyyy-mm-dd.

    By default, the value of this parameter is one day before the scheduled time to run a node. You can customize a time format for this parameter, for example, yyyy-mm-dd for ${yyyy-mm-dd}.

- Examples:
  - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-mm-dd}. If the node is run on July 22, 2018, ${yyyy-mm-dd} is replaced with pt=2018-07-21.
  - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymmdd-2}. If the node is run on July 22, 2018, ${yyyymmdd-2} is replaced with pt=20180719.
  - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyymm-2}. If the node is run on July 22, 2018, ${yyyymm-2} is replaced with pt=201805.
  - The code of an ODPS SQL node includes pt=${datetime}, and the parameter configured for the node is datetime=${yyyy-2}. If the node is run on July 22, 2018, ${yyyy-2} is replaced with pt=2016.
  - You can assign values to multiple parameters when configuring an ODPS SQL node. For example, set startdatetime=$bizdate enddatetime=${yyyymmdd+1} starttime=${yyyy-mm-dd} endtime=${yyyy-mm-dd+1}.

FAQ

- Q: The table partition format is pt=yyyy-mm-dd hh24:mi:ss, but spaces are not allowed in scheduling parameters. How can I configure the format of $[yyyy-mm-dd hh24:mi:ss]?

  A: Use the custom variables datetime=$[yyyy-mm-dd] and hour=$[hh24:mi:ss] to obtain the date and time. Then, join them together to form pt="${datetime} ${hour}" in the code. Separate the two variables with a space.

- Q: The table partition is pt="${datetime} ${hour}" in the code. To obtain the data for the last hour when the node is run, the custom variables datetime=$[yyyymmdd] and hour=$[hh24-1/24] can be used to obtain the date and time, respectively. However, for an instance running at 00:00, it analyzes data for 23:00 of the current day, instead of 23:00 of the previous day. What measures can I take in this case?

  A: Modify the formula of datetime to $[yyyymmdd-1/24] and keep the formula of hour unchanged at $[hh24-1/24]. Then, the node can be run properly.

  - For an instance that is scheduled to run at 2015-10-27 00:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151026 and 23, respectively. This is because the scheduled time minus 1 hour is a time value that belongs to yesterday.
  - For an instance that is scheduled to run at 2015-10-27 01:00:00, the values of $[yyyymmdd-1/24] and $[hh24-1/24] are 20151027 and 00, respectively. This is because the scheduled time minus 1 hour is a time value that belongs to the current day.

DataWorks offers the following node running modes:

- Manually run a node in DataStudio: You must assign temporary values to parameters in the code to ensure proper running of the node. The assigned values are not saved as node properties and do not take effect in other node running modes.
- Automatically run a node at specified intervals: No configuration is needed in the Arguments field. The scheduling system automatically replaces the values of parameters based on the scheduled time of the current instance.
- Test a node or generate retroactive data: You must specify the data timestamp. The scheduled time of each instance can be calculated based on the formula described earlier in this topic.

2.5.6.3. Scheduling properties

This topic describes how to configure the scheduling properties of a node, including the recurrence and dependencies.

You can click the **Properties** tab in the right-side navigation pane of the node configuration tab and set the parameters in the **Schedule** section.

Node status

- **Normal:** If you select this option, the node is run based on the recurrence. By default, this option is selected for a node.
- **Dry Run:** If you select this option, the node is run based on the recurrence. However, the scheduling system does not actually run the code but directly returns a success response.
- **Retry Upon Error:** If you select this check box, the node is rerun when it encounters an error. By default, a node can be automatically rerun for a maximum of three times at an interval of 2 minutes.
- **Skip Execution:** If you select this check box, the node is run based on the recurrence. However, the scheduling system does not actually run the code but directly returns a failure response. You can select this check box if you want to suspend a node and run it later.

Recurrence

After a node is committed and deployed, the scheduling system generates instances every day from the next day based on the scheduling properties of the node. Then, the scheduling system runs the instances based on the running results of ancestor instances and the scheduled time. If a node is committed and deployed after 23:30, the scheduling system generates instances for it from the third day.

> **Note**
>
> If you schedule a node to run every Monday, the node is run only on Mondays. On the other days, the scheduling system does not actually run the code but directly returns a success response. When you test a node scheduled by week or generate retroactive data for the node, you must set the data timestamp to one day earlier than the scheduled time to run the node.
>
> For an auto triggered node, its dependencies take priority over other scheduling properties. That is, when the scheduled time arrives, the scheduling system does not immediately run a node instance but first checks whether all the ancestor instances are run.
>
> - The node instance is in the Not Running state if any ancestor instances are not run when the scheduled time arrives.
> - The node instance is in the Pending (Schedule) state if the scheduled time does not arrive but all the ancestor instances are run.
> - The node instance is in the Pending (Resources) state if all the ancestor instances are run and the scheduled time arrives.

Cross-cycle dependencies

DataWorks supports the following three types of cross-cycle dependencies:

- Dependency on instances of child nodes
  - Node dependency: The current node depends on the last-cycle instances of its child nodes. For example, Node A has three child nodes B, C, and D. If you select this node dependency, Node A depends on the last-cycle instances of nodes B, C, and D.
  - Business scenario: The current node depends on instances of child nodes in the last cycle to cleanse the output tables of the current node and check whether the final result is generated properly.
- Dependency on instances of the current node
  - Node dependency: The current node depends on its last-cycle instances.
  - Business scenario: The current node depends on the data output result of its last-cycle instances.
- Dependency on instances of custom nodes: If you select this node dependency, enter the IDs of the nodes on which the current node depends. You can specify multiple nodes and separate their IDs with commas (,). For example, enter 12345,23456.
  - Node dependency: The current node depends on the last-cycle instances of custom nodes.
  - Business scenario: In the business logic, the current node depends on the proper output of other business data that is not processed by the current node.

> **Note** The difference between cross-cycle dependencies and dependencies in the current cycle lies in that cross-cycle dependencies are displayed as dotted lines in Operation Center.
>
> Before deleting a node from Operation Center, you must delete all dependencies of the node so that other nodes can run properly.

Scheduled by day

Nodes scheduled by day are automatically run once per day. When you create an auto triggered node, the node is set to run at 00:00 every day by default. You can specify another time as needed. In the example shown in the following figure, the time is specified as 13:00.

- If you select Customize Runtime, the node is run at the specified time every day. The time format is YYYY-MM-DD HH:MM:SS.

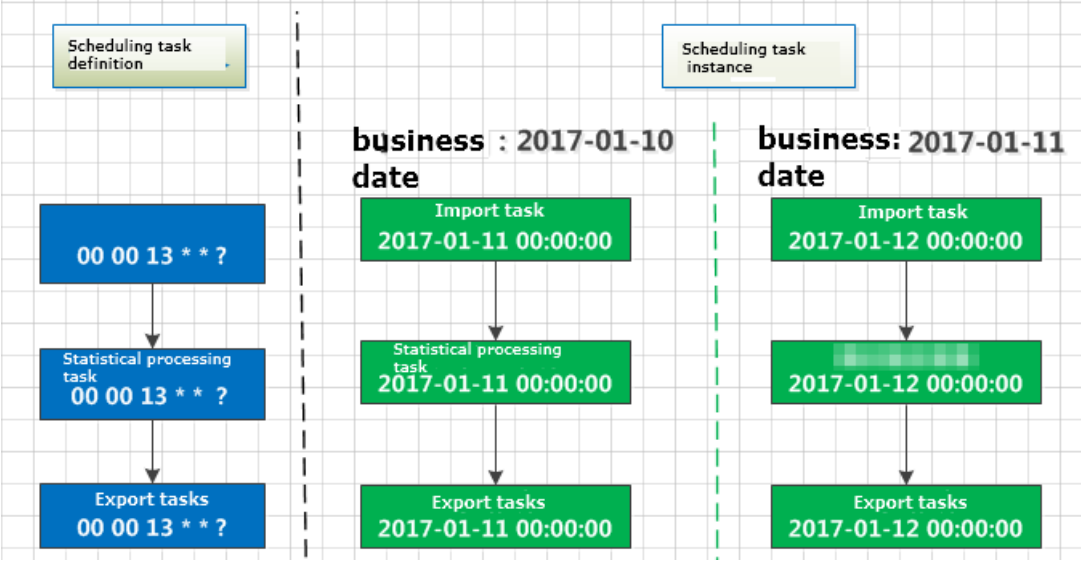
  > **Note** An auto triggered node can be run only when all the ancestor instances are run and the scheduled time arrives. Both prerequisites are indispensable and have no specific chronological order.

- If you clear Customize Runtime, the scheduled time of the node is randomly set in the range of 00:00 to 00:30.

Scenarios:

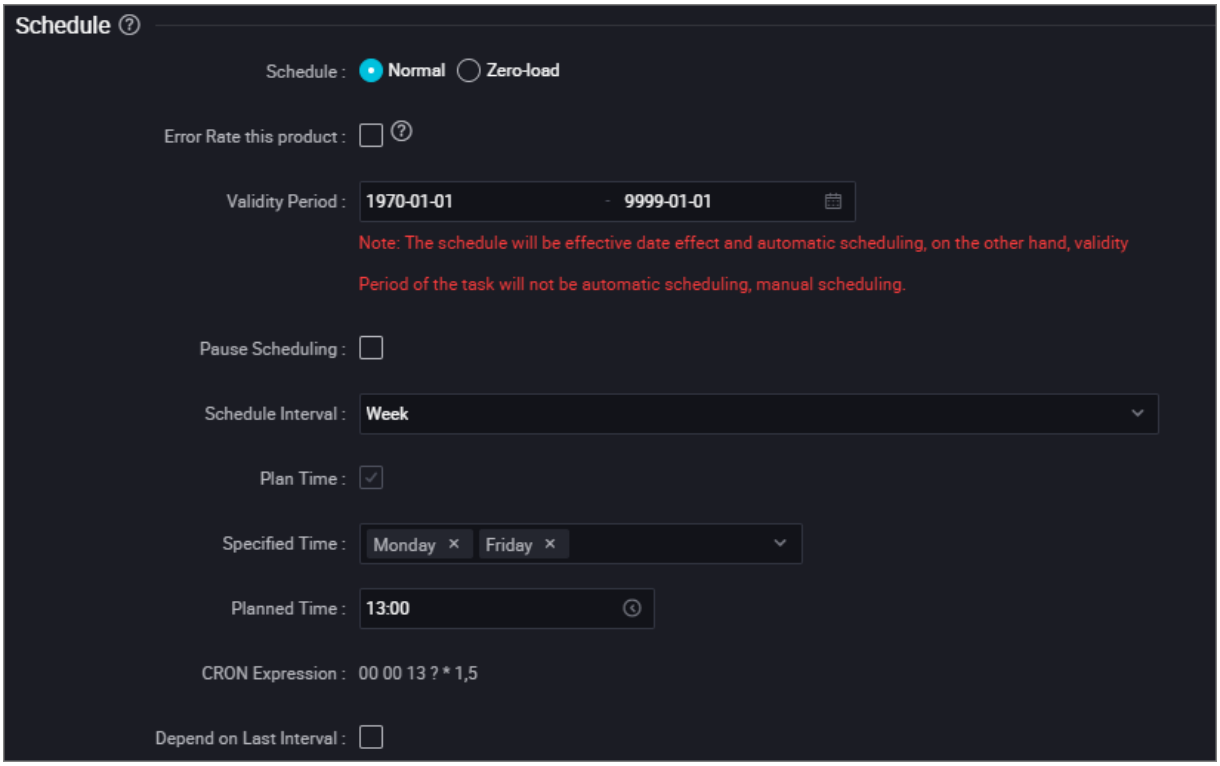
For example, you have created an import node, an analytics node, and an export node. They are all scheduled to run at 13:00 every day. The analytics node depends on the import node, and the export node depends on the analytics node. The following figure shows that the analytics node is configured to depend on the import node.

Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the nodes, as shown in the following figure.



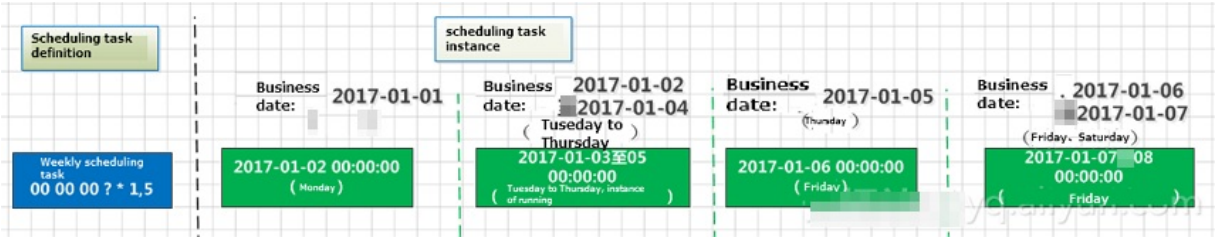
Scheduled by week

Nodes scheduled by week are automatically run at a specified time of specified days every week. On the other days, the scheduling system still generates instances to make sure the proper running of descendant instances. However, the system does not actually run the code or consume resources but directly returns a success response.



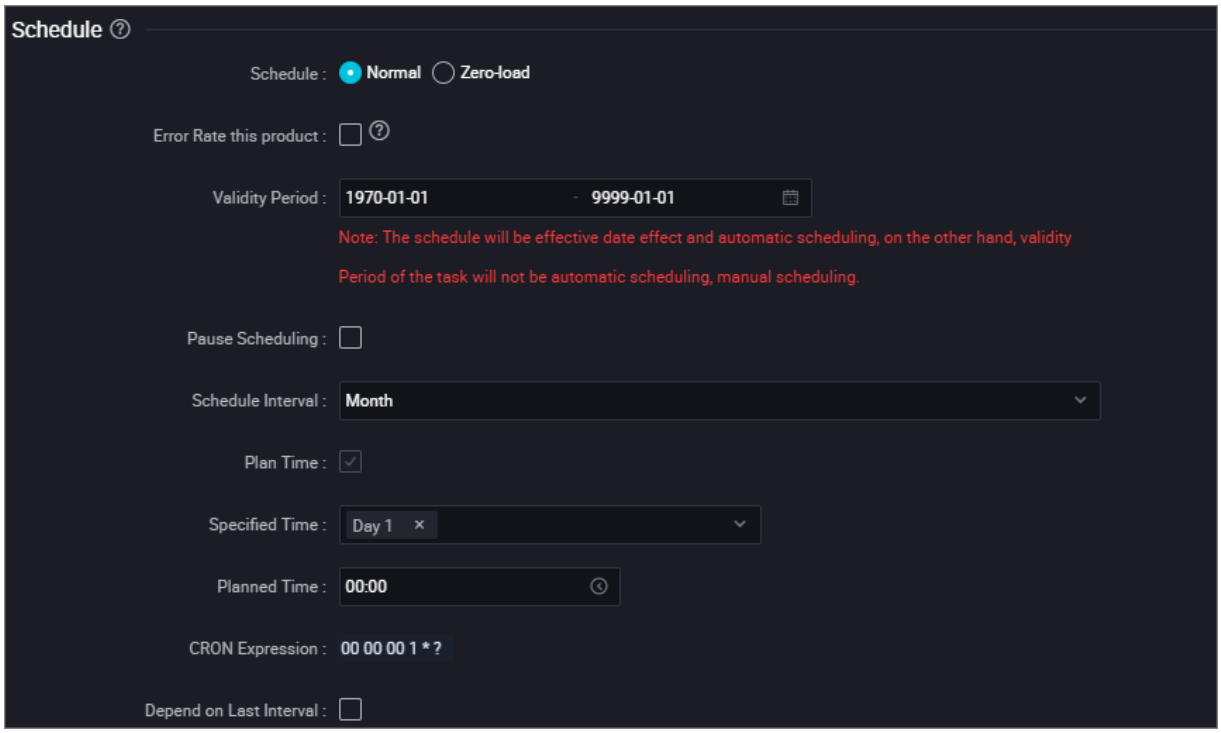
For example, you have created a node. As shown in the preceding figure, the scheduling system runs instances generated on Mondays and Fridays, but returns success responses without running the code for instances generated on Tuesdays, Wednesdays, Thursdays, Saturdays, and Sundays.

Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



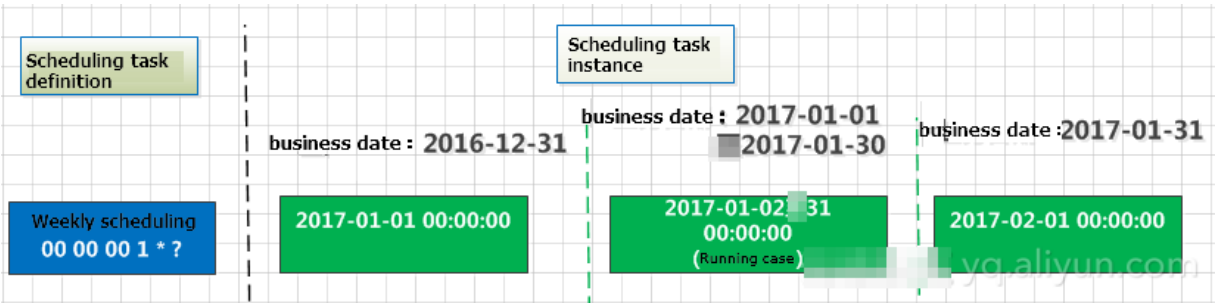
Scheduled by month

Nodes scheduled by month are automatically run at a specified time of specified days every month. On the other days, the scheduling system still generates instances to make sure the proper running of descendant instances. However, the system does not actually run the code or consume resources but directly returns a success response.



For example, you have created a node. As shown in the preceding figure, the scheduling system runs the instance generated on the first day of each month, but returns success responses without running the code for instances generated on the other days.

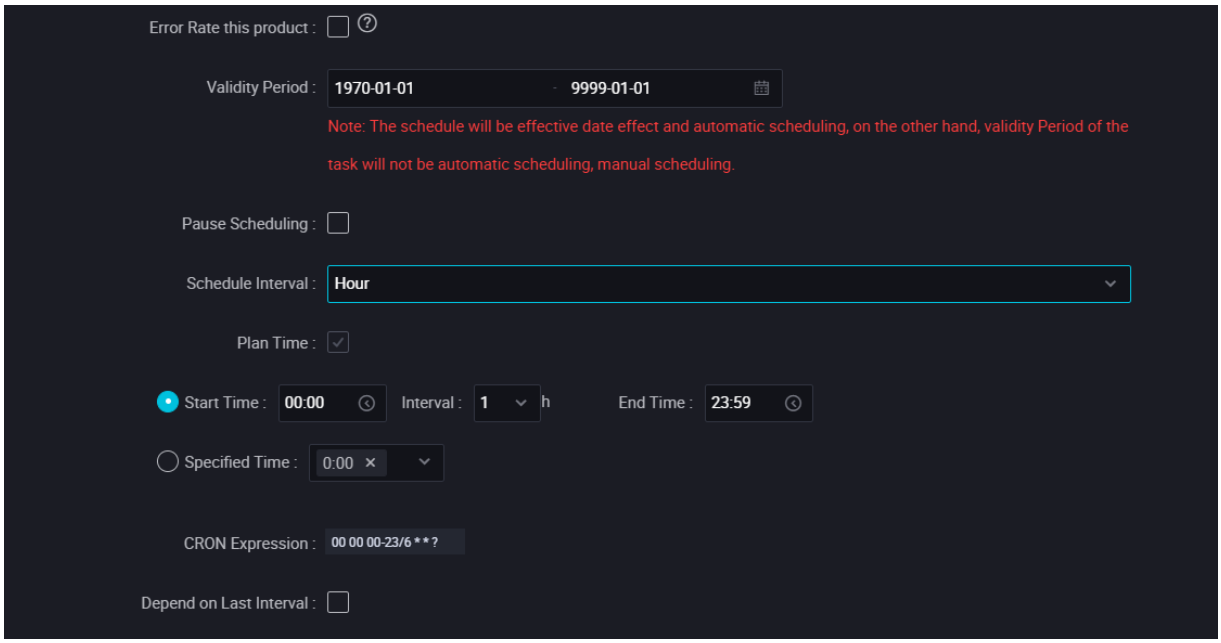
Based on the preceding node scheduling properties, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



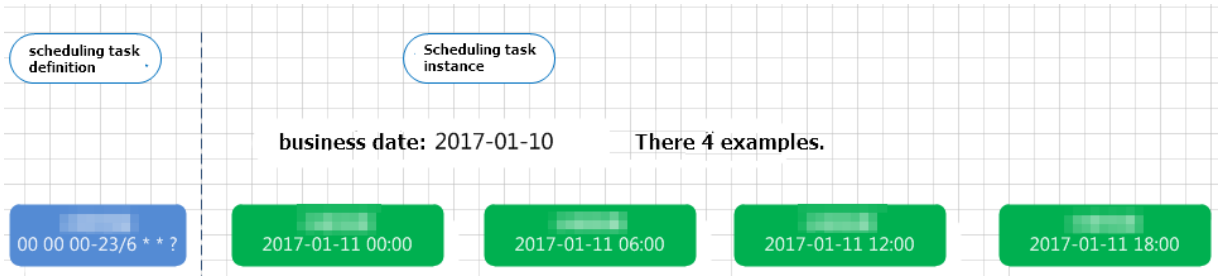
Scheduled by hour

Nodes scheduled by hour are automatically run once every N hours in a specific time period every day. For example, a node is run once per hour from 01:00 to 04:00 every day.

> **Note** The time period is a closed interval. For example, if a node is scheduled to run once per hour in the period from 00:00 to 03:00, the scheduling system generates four instances every day, which are run at 00:00, 01:00, 02:00, and 03:00, respectively.



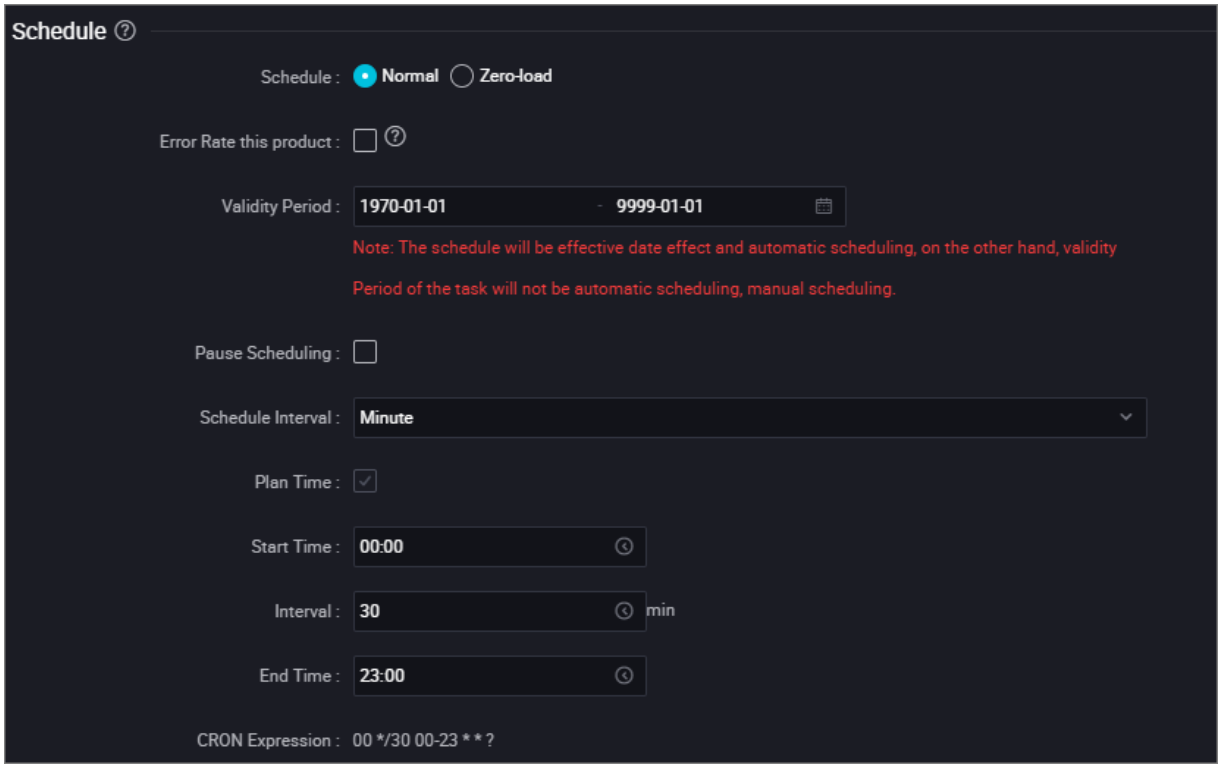
For example, you have created a node. As shown in the preceding figure, the node is automatically run every 6 hours in the period from 00:00 to 23:59 every day. In this case, the scheduling system automatically generates and runs instances for the node, as shown in the following figure.



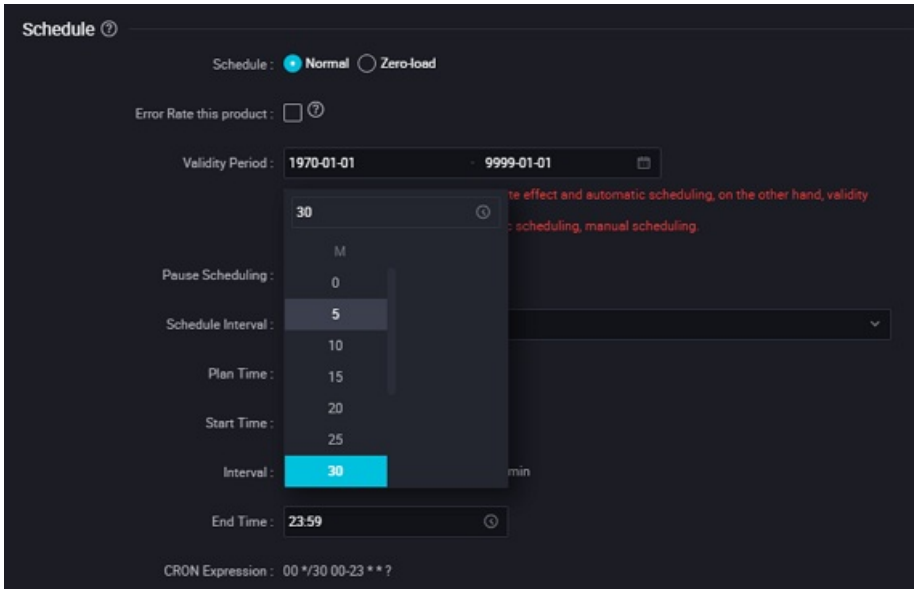
Scheduled by minute

Nodes scheduled by minute are automatically run once every N minutes in a specific time period every day.

For example, you have created a node. As shown in the following figure, the node is run every 30 minutes in the period from 00:00 to 23:00 every day.



Currently, a minimum interval of 5 minutes is supported. The time expression is automatically generated based on the time you select and cannot be modified.



FAQ

- Q: Node A is scheduled by hour and Node B is scheduled by day. How do I enable Node B to automatically run every day after all instances of Node A are run?

A: A node scheduled by day can depend on a node scheduled by hour. To enable Node B to automatically run every day after all 24 instances of Node A are run, do not specify the time to run Node B every day. Then, configure Node A as an ancestor of Node B. For more information, see the Dependencies topic. A node can depend on any other node, regardless of the recurrence. The recurrence of each node is specified in its scheduling properties.

- Q: Node A is run once per hour on the hour every day and Node B is run once per day. How do I enable Node B to automatically run after Node A is run for the first time every day?

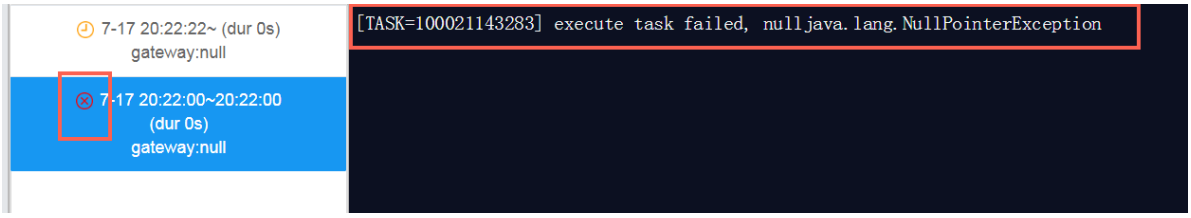
  A: When configuring the scheduling properties of Node A, select **Cross-Cycle Dependencies** and select **Instances of Current Node** from the Depend On drop-down list. When configuring the scheduling properties of Node B, configure Node B to depend on Node A and set the scheduled time of Node B to 00:00 every day. In this way, instances of Node B only depend on the instance of Node A generated at 00:00 every day, that is, the first instance of Node A.

- Q: Node A is run once every Monday and Node B depends on Node A. How do I enable Node B to run once every Monday?

  A: Set the scheduling properties of Node B to be the same as those of Node A. That is, select **Week** as the instance recurrence and select **Monday**.

- Q: How are the instances of a node affected when the node is deleted?

  A: When a node is deleted, its instances are retained because the scheduling system still generates one or more instances for the node based on the scheduling properties. However, when the scheduling system runs such instances after the node is deleted, an error message appears because the required code is unavailable, as shown in the following figure.



- Q: Can I enable a node to process monthly data on the last day of each month?

  A: No, DataWorks does not support setting a node to run on the last day of each month. If you enable a node to run on the thirty-first day of each month, the scheduling system runs a node instance in each month that has 31 days and returns a success response without running the code in any other month.

  We recommend that you configure a node to process the data of the past month on the first day of each month.

- Q: If a node scheduled by day depends on a node scheduled by hour, how do I enable the node scheduled by day to run at 00:00 every day?

  A: You can configure the node scheduled by day to depend on the data generated on the day before for the node scheduled by hour. If the node scheduled by day depends on the data generated on the current day for the node scheduled by hour, the instances of the node scheduled by day can be run only on the next day.

  In the Schedule section of the node scheduled by day, select **Cross-Cycle Dependencies**, select **Instances of Custom Nodes** from the Depend On drop-down list, and then enter the ID of the node scheduled by hour on which the node scheduled by day depends. Commit and deploy the node scheduled by day.

- Q: What can I do if I do not know when the output data of the ancestor node is generated?

A: You can set the cross-cycle dependency for the current node to depend on the last-cycle instances of the ancestor node.

- Q: After a modified node is committed and deployed to the production environment, will the node instances that were originally faulty in the production environment be overwritten?

  A: No, the node instances that have been generated will not be overwritten. The updated code is used to run the node instances that are newly generated and have not been run. If the scheduling properties are modified, the modified configuration also applies to the newly generated node instances.

2.5.6.4. Dependencies

Scheduling dependencies are the foundation for building orderly workflows. You must configure correct dependencies between nodes to make sure that business data is produced effectively and in a timely manner. This helps standardize data development activities.

DataWorks allows you to automatically parse node dependencies from the code or manually customize node dependencies. You can configure correct relationships between ancestor and descendant nodes and monitor the running status of nodes to make sure the orderly production of business data.

The purpose of configuring node dependencies is to check the data output time of the table queried by SQL statements and check whether data is properly produced from an ancestor node based on the node status.

You can set the output of an ancestor node as the input of a descendant node to configure a dependency between the two nodes.

Regardless of the dependency configuration mode, the overall scheduling logic is that descendant nodes can be run only after ancestor nodes are run. Therefore, each node in a workflow must have at least one parent node. The dependencies between the parent nodes and child nodes are the core of scheduling dependencies. The following sections describe the principles and configuration methods of scheduling dependencies in detail.

Differences between automatic parsing and custom dependencies

DataWorks can automatically parse the input and output of a node based on the lineage parsed from the code.

If the lineage parsed from the code is inaccurate, you can add custom dependencies as needed. We recommend that you write the code correctly to parse the lineage from the code and reduce custom dependencies. The following example shows how to configure the input and output of a node.

Auto Parse: If you select Yes, node dependencies are automatically parsed from the code.

For example, the code of an ODPS SQL node is as follows:

```
insert overwrite table table_a as select * from project_b_name.table_b;
```

From the code, DataWorks determines that the current node depends on the node that generates table_b and the current node generates table_a. Therefore, the output name of the parent node is project_b_name.table_b and the output name of the current node is project_name.table_a.

- If you do not want to parse node dependencies from the code, select **No** for Auto Parse.
- If a table in an SQL statement is both an output table and a referenced table on which another node depends, the table is parsed only as an output table.
- If a table in an SQL statement is used as an output table or a referenced table multiple times, only one scheduling dependency is parsed.
- If the SQL code contains a temporary table, the table is not involved in a scheduling dependency. Temporary tables are prefixed with t_. For more information, see Project Configuration.

Parent nodes

In the Dependencies section of a node, you must specify an ancestor node as the parent node on which the current node depends. You must enter the output name of the ancestor node, rather than the ancestor node name. A node may have multiple output names. Enter an output name as needed. You can search for an output name of the ancestor node to be added, or click Parse I/O to parse the output name based on the lineage parsed from the code.

> **Note** You must enter an output name or output table name to search for the ancestor node.
>
> If you enter an output name to search for the ancestor node, DataWorks searches for the output name among the output names of nodes that have been committed to the scheduling system.

- Search by entering an output name

  You can enter an output name to search for the ancestor node and configure the node as the parent node of the current node to create a dependency.

- Search by entering an output table name

  When using this method, make sure that the entered output table name of the ancestor node is the table name used in the INSERT or CREATE statement of the current node, such as Project name.Table name . Such output names can be automatically parsed.

  After you click the **Submit** icon, the output table name of the parent node configured for the current node can be found when you enter an output table name to search for the ancestor node for other nodes.

Outputs

You can click the **Properties** tab in the right-side navigation pane to view and configure the output of the current node.

DataWorks assigns a default output name that ends with .out to each node. You can also customize an output name or click Parse I/O to parse the output name based on the lineage parsed from the code.

> **Note** The output name of each node must be globally unique.

FAQ

- Q: After DataWorks automatically parses the input and output of a node, the node fails to be committed. An error message appears to indicate that the parsed output name

workshop_yanshi.tb_2 of the parent node does not exist and you must commit the parent node before committing the current node. Why does this error occur?

A: The possible causes are as follows:

- The ancestor node is not committed. Commit the ancestor node and try again.
- The ancestor node is committed, but workshop_yanshi.tb_2 is not an output name of the ancestor node.

> **Note** Usually, the output names of the parent node and the current node are automatically parsed based on the table name that is used in the INSERT or CREATE statement or follows the FROM keyword. Make sure that you follow the principles of automatic parsing in the Differences between automatic parsing and custom dependencies section.

- Q: In the output of the current node, the descendant node name and ID are empty and cannot be specified. Why does this happen?

  A: If the current node does not have a descendant node, the descendant node name and ID are empty. After a descendant node is configured for the current node, the corresponding content can be automatically parsed.

- Q: What is the output name of a node used for?

  A: The output name of a node is used to establish dependencies between nodes. For example, if the output name of Node A is ABC and Node B uses ABC as its input, a dependency is established between nodes A and B.

- Q: Can a node have multiple output names?

  A: Yes, a node can have multiple output names. If a descendant node references an output name of the current node as the output name of the parent node, a dependency is established between the descendant node and the current node.

- Q: Can multiple nodes have the same output name?

  A: No, the output name of each node must be unique under your Apsara Stack tenant account. If multiple nodes export data to the same MaxCompute table, we recommend that you use Table name_Partition ID as the output name format of these nodes.

- Q: How can I avoid intermediate tables when I enable DataWorks to automatically parse node dependencies?

  A: Right-click an intermediate table name in the SQL code and select **Delete Input** or **Delete Output**. Then, click Parse I/O to parse the input and output of the node.

- Q: How do I configure dependencies for the upmost node in a workflow?

  A: You can set the node to depend on the root node of the current workspace.

- Q: Why do I find a non-existent output name of Node B when I enter an output name to search for the ancestor node for Node A?

  A: DataWorks searches for the output name among the output names of nodes that have been committed to the scheduling system. After Node B is committed, if you delete the output name of Node B and does not commit Node B to the scheduling system again, the deleted output name of Node B can still be found.

- Q: How do I enable nodes A, B, and C to run in sequence once per hour?

A: Set the output of Node A as the input of Node B and the output of Node B as the input of Node C. Also, set nodes A, B, and C to run once per hour.

- Q: An error message is returned to indicate that the parent node ID fails to be automatically parsed based on an output table name. Why does this error occur?

  A: This error does not indicate that the table does not exist. Instead, it indicates that the table is not the output of a specific node. Therefore, the table name cannot be used to find the node that generates the table data. In this case, the dependency on the node cannot be created.

  According to the principles of automatic parsing described in this topic, a dependency is created after the output of an ancestor node is set as the input of a descendant node. If no ancestor node can be parsed based on the xc_demo_partition table referenced in SQL statements, no node uses the xc_demo_partition table as its output.

  You can resolve this problem in the following way:

  i. Find the node that generates the table data and view the node output.

     If you do not know which the target node is, you can enter keywords to search the code for the node in fuzzy match mode.

  ii. If the table data is uploaded from a local server or you do not need to depend on the node, you can right-click the table name in the code and select **Delete Input**.

  > **Note** We recommend that you write the code correctly to parse the lineage from the code and reduce custom dependencies.

2.5.7. Components

2.5.7.1. Create a script template

This topic describes the definition and composition of script templates and how to create a script template.

Definition

A script template defines an SQL code process that involves multiple input and output parameters. Each SQL code process references one or more source tables. You can filter source table data, join source tables, and aggregate them to generate a result table required for new business.

Value

In actual business, many SQL code processes are similar. The input and output tables in these processes may have the same or compatible schema but different names. In this case, developers can abstract an SQL code process as a script template to reuse the SQL code. The script template extracts input parameters from input tables and generates output parameters in output tables.

To create SQL script templates, you can select script templates from the script template list based on your business process and configure specific input and output tables in your business for the selected script templates, without repeatedly copying the code. This greatly improves the development efficiency and avoids repeated development. You can deploy and run the created SQL script templates in the same way as other SQL nodes.

Composition

Similar to a function, a script template consists of input parameters, output parameters, and an SQL code process.

Input parameters

The input parameters of a script template have the properties such as the parameter name, parameter type, parameter description, and parameter definition. The parameter type can be table or string.

- A table-type parameter specifies the table to be referenced in an SQL code process. When you use a script template, you can specify the input table required for the specific business.
- A string-type parameter specifies the variable control parameter in an SQL code process. For example, to export only the sales amount of the top N cities in each region in a result table of an SQL code process, you can use a string-type parameter to specify the value of N.

  To export the total sales amount of a province in a result table of an SQL code process, you can set a string-type parameter to specify the province and obtain the sales data of the specified province.
- The parameter description specifies the role of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema, which is required only for table-type parameters. When you specify the parameter definition for a table-type parameter, you must provide an input table that contains the same field names and compatible types defined by the table-type parameter so that the SQL code process can run properly. Otherwise, an error is returned when the SQL code process runs because the specified field name cannot be found in the input table. The input table must contain the field names and types defined by the table-type parameter. The input table can also contain other fields. The field names and types in the input table can be in any order. The parameter definition is for reference only.
- We recommend that you enter the parameter definition in the following format:

  ```
  Name of field 1 Type of field 1 Description of field 1
  Name of field 2 Type of field 2 Description of field 2
  Name of field n Type of field n Description of field n
  ```

  Examples:

  ```
  area_id string 'Region ID'
  city_id string 'City ID'
  order_amt double 'Order amount'
  ```

Output parameters

- The output parameters of a script template have the properties such as the parameter name,

parameter type, parameter description, and parameter definition. The parameter type must be table. A string-type output parameter has no logical meaning.

- A table-type parameter specifies the table to be generated in an SQL code process. When you use a script template, you can specify the result table that the SQL code process generates for the specific business.
- The parameter description specifies the role of a parameter in an SQL code process.
- The parameter definition is a text definition of the table schema. When you specify the parameter definition for a table-type parameter, you must provide an output table that contains the same number of fields and compatible types defined by the table-type parameter so that the SQL code process can run properly. Otherwise, an error is returned when the SQL code process runs because the number of fields does not match or the field type is incompatible. The field names of the output table do not need to be consistent with those defined by the table-type parameter. The parameter definition is for reference only.
- We recommend that you enter the parameter definition in the following format:

```
Name of field 1 Type of field 1 Description of field 1
Name of field 2 Type of field 2 Description of field 2
Name of field n Type of field n Description of field n
```

Examples:

```
area_id string 'Region ID'
city_id string 'City ID'
order_amt double 'Order amount'
rank bigint 'Ranking'
```

SQL code process

The parameters in an SQL code process are referenced in the following format: `@@{Parameter name}` .

By containing an abstract SQL code process, a script template controls and processes an input table based on input parameters to generate an output table with business value.

To develop an SQL code process, you must use input and output parameters in the code properly to make sure that they can be set as needed and correct SQL code can be generated and run during the process.

Create a script template

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Snippets** icon.
3. On the Snippets tab, move the pointer over and choose **Create** > **Snippet**.
4. In the **Create Snippet** dialog box, set **Snippet Name**, **Description**, and **Location**.
5. Click **Commit**.

Source table schema

The following table describes the schema of a source MySQL table that contains sales data.

Field	Data type	Description
order_id	varchar	The ID of the order.
report_date	datetime	The date of the order.
customer_name	varchar	The name of the customer.
order_level	varchar	The level of the order.
order_number	double	The number of orders.
order_amt	double	The amount of the order.
back_point	double	The discount.
shipping_type	varchar	The transportation method.
profit_amt	double	The amount of the profit.
price	double	The unit price.
shipping_cost	double	The transportation cost.
area	varchar	The region.
province	varchar	The province.
city	varchar	The city.
product_type	varchar	The type of the product.
product_sub_type	varchar	The subtype of the product.
product_name	varchar	The name of the product.
product_box	varchar	The packaging of the product.
shipping_date	datetime	The shipping date.

Business implication

Script template name: get_top_n

This script template uses the specified sales data table as the table-type input parameter, the number of the top cities as the string-type input parameter, and the total sales amount of the cities for ranking. By using this SQL code process, you can obtain the rankings of the specified top cities in each region with ease.

Script template parameters

Input parameter 1

- Parameter name: myinputtable
- Type: table

Input parameter 2

- Parameter name: topn
- Type: string

Output parameter 3

- Parameter name: myoutput
- Type: table

Parameter definition:

- area_id string
- city_id string
- order_amt double
- rank bigint

You can execute the following statement to create a table for storing the sales data of a specified number of top cities:

```
CREATE TABLE IF NOT EXISTS company_sales_top_n
(
area STRING COMMENT 'Region',
city STRING COMMENT 'City',
sales_amount DOUBLE COMMENT 'Sales amount',
rank BIGINT COMMENT 'Ranking'
)
COMMENT 'Company sales rankings'
PARTITIONED BY (pt STRING COMMENT '')
LIFECYCLE 365;
```

Example of defining an SQL code process

```
INSERT OVERWRITE TABLE @@{myoutput} PARTITION (pt='${bizdate}')
   SELECT r3.area_id,
   r3.city_id,
   r3.order_amt,
   r3.rank
from (
SELECT
   area_id,
   city_id,
   rank,
   order_amt_1505468133993_sum as order_amt ,
   order_number_150546813****_sum,
   profit_amt_15054681****_sum
FROM
```

```
   (SELECT
   area_id,
   city_id,
   ROW_NUMBER() OVER (PARTITION BY r1.area_id ORDER BY r1.order_amt_1505468133993_sum DESC)
AS rank,
   order_amt_15054681****_sum,
   order_number_15054681****sum,
   profit_amt_1505468****_sum
FROM
   (SELECT area AS area_id,
    city AS city_id,
    SUM(order_amt) AS order_amt_1505468****_sum,
    SUM(order_number) AS order_number_15054681****_sum,
    SUM(profit_amt) AS profit_amt_1505468****_sum
FROM
   @@{myinputtable}
WHERE
   SUBSTR(pt, 1, 8) IN ( '${bizdate}' )
GROUP BY
   area,
   city )
   r1 ) r2
WHERE
   r2.rank >= 1 AND r2.rank <= @@{topn}
ORDER BY
   area_id,
   rank limit 10000) r3;
```

Sharing scope

Script templates can be shared within a workspace or made public.

By default, a deployed script template is visible and available to users within the current workspace. The developer of a script template can click the **Publish Snippet** icon to make the general-purpose script template public to the current tenant account so that all users under the account can view and use the script template.

You can view the **Publish Snippet** icon in the toolbar of the configuration tab of a script template. If the icon is clickable, the script template is made public.

Use of script templates

For more information about how to use a developed script template, see Use components.

Reference records

In the script template list, double-click a script template. On the configuration tab that appears, click the **Snippet Nodes** tab in the right-side navigation pane to view the reference records of the script template.

2.5.7.2. Use a script template

To improve development efficiency, you can create data analytics nodes by using the script templates provided by workspace members and tenants.

Note the following points when you use script templates:

- The script templates provided by members of the current workspace are available on the **Workspace-Specific** tab.
- The script templates provided by tenants are available on the **Public** tab.

GUI elements

Icon or tab	Description
Save icon	Saves the settings of the current script template.
Steal Lock icon	Allows you to steal the lock of the current script template and then edit it if you are not the owner of the script template.
Submit icon	Commits the current script template to the development environment.
Publish Snippet icon	Makes the current general-purpose script template public to the current tenant account so that all users under the account can view and use the script template.
Parse I/O Parameters icon	Parses input and output parameters from the code. **Note** Typically, the parameters entered here are table names instead of scheduling parameters.
Run icon	Runs the current script template in the development environment.
Stop icon	Stops running the current script template.
Format icon	Formats the code based on keywords.
Parameters tab	Allows you to view the basic information and set input and output parameters for the current script template.
Versions tab	Allows you to view the deployed versions of the current script template.
Snippet Nodes tab	Lists the reference records of the current script template.

2.5.8. Custom node type

2.5.8.1. Overview

DataStudio supports default node types such as ODPS SQL and Shell nodes. You can also create custom node types to meet your requirements.

To create a custom node type, you need to create a custom wrapper and use it to define a custom node type.

Entry

1. Log on to the DataWorks console.
2. Click **Node Market** in the upper-right corner to go to the node configuration page.

> **Note** Only the workspace owner and administrators can access this page.

View the list of wrappers

The Wrappers page displays all the wrappers you have created. You can click **Create** in the upper-right corner to create a custom wrapper.

The values displayed in the Latest Version, Version in Development, and Version in Production Environment columns for the created wrappers follow these rules:

- If a created wrapper has not been deployed, the values of both the Version in Development and Version in Production Environment columns are **Not Deployed**.
- If a wrapper has been deployed, the version and the deployment time appear in these columns.
- If a wrapper is under deployment, the values of both the Version in Development and Version in Production Environment columns are **Deploying**.

You can click **Settings**, **View Versions**, or **Delete** in the Actions column of each wrapper.

Action	Description
Settings	You can click **Settings** to configure the wrapper. The page that appears depends on the wrapper status. The **Deploy in Production Environment** page appears if the wrapper has been deployed in the production environment.
View Versions	You can click **View Versions** to view all historical versions of the wrapper. • **View**: You can click this button to view the settings of the selected version. • **Roll Back**: You can click this button to roll back to the selected version. After you click this button, the system creates a new version for the wrapper. In the new version, the wrapper uses the basic settings and the resource file of the selected version. The new version number equals the latest version number among all the versions plus 1. • **Download**: You can click **Download** to download the resource file of the selected version.

Action	Description
Delete	If an error occurs while a node type is using the wrapper, you need to delete the node type. Note Before deleting a wrapper, ensure that no node type is associated with the wrapper.

Create a custom wrapper

A wrapper is the core processing logic of a node type. For example, after you compile an SQL statement in the editor for an ODPS SQL node and submit the statement, the system calls the corresponding wrapper to parse and run the statement. You need to create a wrapper before creating a custom node type. Currently, only the Java programming language is supported.

The procedure of creating a wrapper includes four steps: specify settings for the wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment. For more information, see Create a custom wrapper.

View the list of custom node types

The Custom Node Types page displays all custom node types in the workspace. You can click **Create** in the upper-right corner to create a custom node type. For more information, see Create a custom node type.

Currently, you can only create custom node types in DataStudio.

The workspace owner or node type creator can **change** or **delete** existing node types.

- **Change**: You can click **Change** to edit the settings for the node type as needed.
- **Delete**: You can click this button to delete the node type that no node uses. If any node uses the node type, a message appears, indicating that you need to disable the node first before deleting the node type.

Use a custom node type

After creating a custom node type, go to the **Data Analytics** page.

Move the pointer over the **Create** icon and click **Data Analytics**. In the list that appears, select the created node type to create a node.

2.5.8.2. Create a custom wrapper

The procedure of creating a wrapper includes four steps: specify settings for a wrapper, deploy the wrapper in the development environment, test the wrapper in the development environment, and deploy the wrapper in the production environment.

Specify settings for a wrapper

1. Click **Wrappers** in the left-side navigation pane. On the page that appears, click **Create** in the upper-right corner.
2. Specify the parameters in the **Settings** step.

Parameter	Description
Name	The name of the wrapper. It must start with a letter and can only contain letters, digits, and underscores (_).
Owner	The owner of the wrapper. You can select an owner from the workspace members. You are not allowed to edit wrappers owned by other members even if you are an administrator. Only the workspace owner can edit the wrappers of other members.
Resource Type	The type of the resource package for configuring the wrapper. Valid values: **JAR** and **Archive**. The size of the resource package can be up to 50 MB.
Resource File	The local resource file or OSS object for configuring the wrapper. **Note** The size of a local file can be up to 50 MB, and the size of a file that is stored in an OSS bucket can be up to 200 MB.
Class Name	The full path of the class for implementing the user wrapper.
Parameter Example	The parameters designed based on the JAR package you upload.
Version	The version of the configured wrapper. Select Create Version if you are creating a new wrapper. Select Overwrite Version if you are editing and rolling back a version. **Note** The version number is automatically generated.
Description	The description of the wrapper version.

3. Click **Save** and then click **Next**.

Note The settings are updated to the database after you click Save.

- If you only modify basic settings of a wrapper without changing the resource file, the modification takes immediate effect after you click Save.
- If you change the resource file, the change only applies after deployment.

Deploy the wrapper in the development environment

After you specify the parameters in the **Settings** step and click **Next**, the information in the Deploy in Development Environment step is updated accordingly. You can identify the changes by checking the file name and MD5 checksum.

Click **Deploy in Development Environment**. You can view the **deployment progress** in real time. After the wrapper is deployed, click **Next**.

Test the wrapper in the development environment

Specify the parameters for testing and click **Test** to send the parameters to the wrapper. This step is to validate the deployment and logic of the wrapper. You can also locally test the wrapper before uploading it for deployment.

After the test, review the output logs in the Test Results section on the right to determine whether the test is passed. If the test is passed, select **Test Passed** and click **Next**.

> **Note** The **Next** button is operable only after you select **Test Passed**.

Deploy the wrapper in the production environment

Click **Deploy in Production Environment**. In the **Confirm** dialog box that appears, click **OK**. The wrapper is deployed in the production environment. You can view the deployment progress in real time.

> **Note** The wrapper to be deployed in the production environment must be of the latest version, have been deployed in the development environment, and have passed the test. Otherwise, a message appears, indicating that the deployment in the production environment fails.

Click **Complete**. You can view and edit the created wrapper on the **Wrappers** page.

2.5.8.3. Create a custom node type

The Configure Custom Node Type page consists of three sections: Basic Information, Interaction, and Wrapper.

1. On the DataStudio page, click **Node Market** in the top navigation bar. On the page that appears, click **Custom Node Types** in the left-side navigation pane.
2. Click **Create** in the upper-right corner.
3. Specify the parameters in the **Basic Information** section.

Parameter	Description
Name	The name of the node type, which cannot be changed after being saved. Each node type has a unique name within the workspace. The name can be up to 20 characters in length, and can only contain letters, spaces, and underscores (_).
Icon	The icon of the node type.
Tabs	The template of the node type. Currently, only **Data Analytics** is available.
Folder	The folder where the node type belongs. You can select **Data Integration** or **Data Analytics**.

4. Specify the parameters in the **Interaction** section.

Parameter	Description

Parameter	Description
Shortcut Menu	○ The options to appear in the shortcut menu. The following options are selected by default: Rename, Move, Clone, Steal Lock, View Versions, Locate in Operation Center, Delete, and Submit for Review. ○ More options include Edit, Copy Resource Name, and Send to DataWorks Desktop (Shortcut).
Tool Bar	○ The options to appear in the top navigation bar. The following options are selected by default: Save, Commit, Commit and Unlock, Steal Lock, Run, Show/Hide, Run with Arguments, Stop, Reload, Run Smoke Test in Development Environment, View Smoke Test Log in Development Environment, Run Smoke Test, View Smoke Test Log, Go to Operation Center of Development Environment, and Format. ○ More options include Operation Center, Deploy, and Precompile.
Editor Type	The type of the editor. Currently, only **Editor Only** is available.
Right-Side Bar	○ The options to appear in the right-side bar. The following options are selected by default: Code Structure and Properties. ○ More options include Version, Lineage, and Parameters.
Auto Parse Option	Specifies whether to display the Auto Parse option for this type of node. If you turn on this switch, the Auto Parse option is displayed on the Properties tab. Otherwise, it is not displayed. In an automatic parsing process, the system parses the input and output of a node based on the lineage specified in the code.

5. Specify the parameters in the **Wrapper** section.

Parameter	Description
Wrapper	The wrapper used for running the type of node. Select a wrapper that has been deployed.
Editor Language	The language used for writing the code in the editor. Currently, only ODPS SQL is available.
Use MaxCompute as Engine	Specifies whether to use MaxCompute as the compute engine. If your wrapper uses MaxCompute as the compute engine, select Yes. Otherwise, select No. Default value: Yes.

6. Click **Save and Exit**. Then, go to the **Data Analytics** page to use the custom node type that is created.

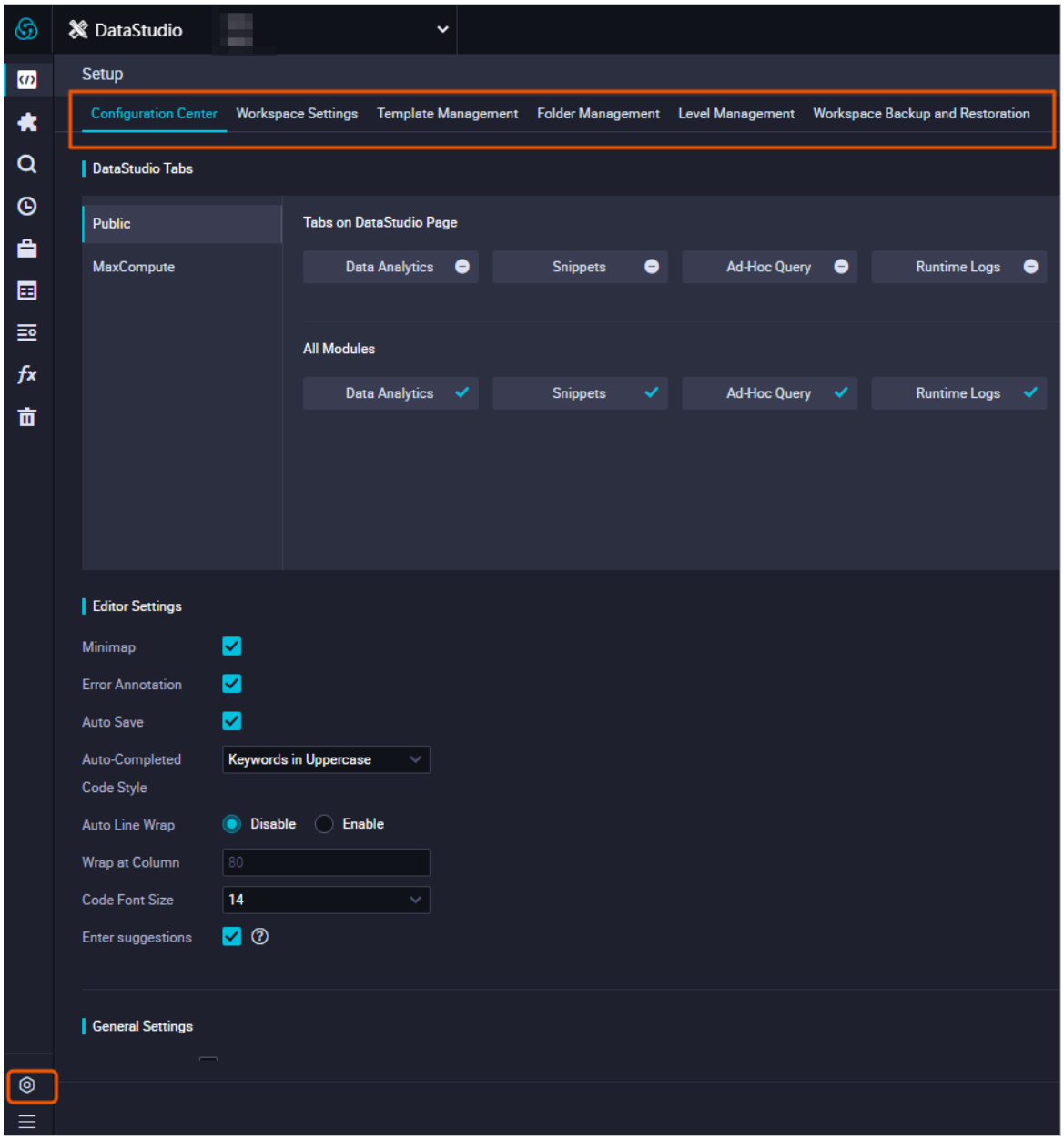
2.5.9. Manage configurations

2.5.9.1. Setup

On the Setup page, you can add and delete modules. You can also configure code templates, folders, and table levels on this page.

Procedure

1. Log on to the DataWorks console.
2. Click in the lower-left corner of the **DataStudio** page to go to the **Setup** page.



You can perform operations on the following tabs:

- Configuration Center

- Project Configuration
- Templates
- Theme Management
- Table Levels

2.5.9.2. Configuration center

You can combine your DataStudio modules and specify editor settings on the Configuration Center tab.

Go to Configuration Center

1. Log on to the DataWorks console.
2. Click in the lower-left corner of the **DataStudio** page. The **Configuration Center** tab appears.

   The Configuration Center tab includes three sections: **DataStudio Tabs**, **Editor Settings**, and **General Settings**. After the configurations are completed, you can click **Apply to All Workspaces** in the lower-right corner of the page to apply the settings to all existing workspaces.

DataStudio tabs

On the **DataStudio Tabs** tab, you can add and delete public and MaxCompute functional modules, and drag modules to change their orders.

- Under **Tabs on DataStudio Page**, click next to a module to delete it. Deleted modules will not appear in the left-side navigation pane of the **DataStudio** page.
- Under **All Modules**, click the desired module to add it. Added modules will appear in the left-side navigation pane of the **DataStudio** page.

> **Note** The module settings take effect immediately for the current workspace. To make the module settings effective for all workspaces, click **Apply to All Workspaces** in the lower-right corner of the page.

Editor settings

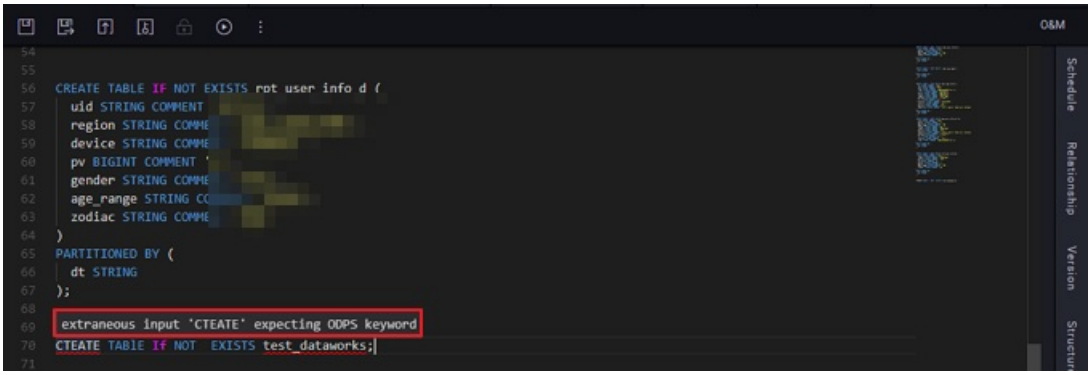
You can configure the code editor in the Editor Settings section. The editor settings take effect immediately for the current workspace without requiring you to refresh the page.

- **Minimap**

The masked code in the current interface is displayed in the minimap in the upper-right corner of the page. When the code is long, you can move the pointer to specify the code block to be displayed in the minimap.

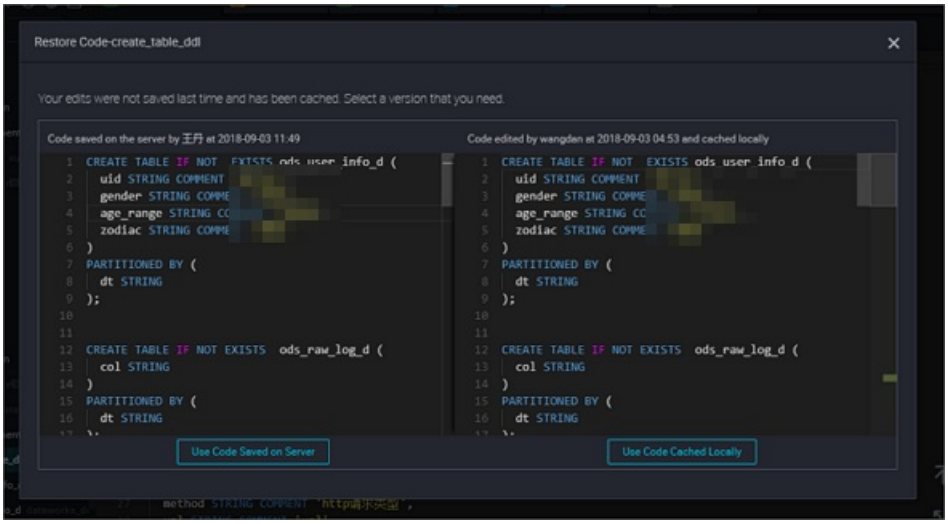
- **Error Annotation**

  If you select this check box, DataWorks marks potential syntax errors with a red squiggly line. When you see a syntax error, you can move the pointer over the underlined code to view the error message.



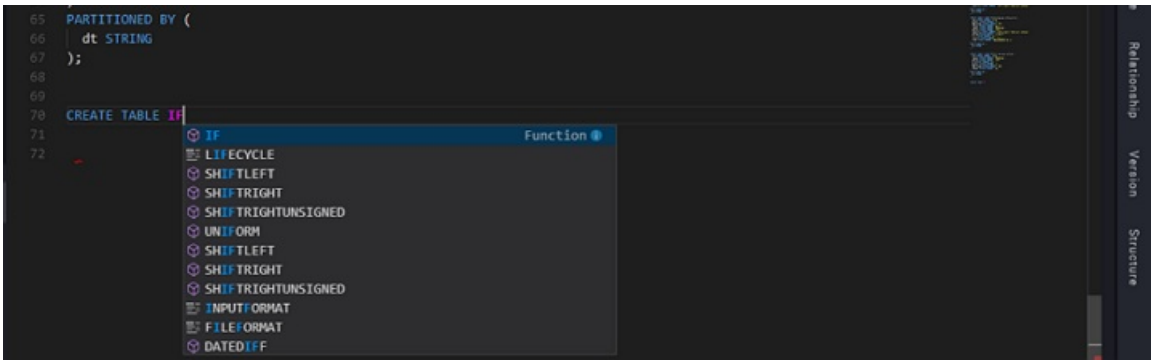
- **Auto Save**

  If you select this check box, DataWorks automatically saves the code being edited at a specific interval. In this way, if the code editor of a node is closed unexpectedly, you can click **Use Version Saved on Server** or **Use Version Saved in Local Cache** to re-open the node.



- **Auto-Completed Code Style**

You can set the code style to uppercase or lowercase as required.



- **Auto Line Wrap**

  You can set Auto Line Wrap to **Disable** or **Enable**.

- **Wrap at Column**
  - If Auto Line Wrap is set to **Disable**, the value of Wrap at Column is 80 by default, which cannot be modified.
  - If Auto Line Wrap is set to **Enable**, you can set a value for Wrap at Column as required.

- **Code Font Size**

  Valid values: 12 to 18. You can change the font size based on your habits and code size.

- **Enter suggestions**

  If you select this check box, the system automatically displays suggestions on how to set a field when you press Enter. If you clear this check box, a new line is started after you press Enter. In addition to the Enter key, you can also use the Tab key to enter prompted suggestions.

- **Auto Completion**

  You can specify whether to enable the following code hints when you enter the code:
  - **Continuous Smart Tips**: specifies whether to automatically add a space after each auto-completed term such as a keyword, table name, or field name.
  - **Keyword**: specifies whether to enable keyword hints.
  - **Syntax Template**: specifies whether to enable syntax template hints.
  - **Project**: specifies whether to enable project name hints.
  - **Table Name**: specifies whether to enable table name hints. When this feature is enabled, the system gives higher priority to tables used recently.
  - **Field**: specifies whether to enable field name hints.

General settings

- **Display Node Engine Information on DAG**

  You can specify whether to display node engine information on the DAG.

- **Theme**

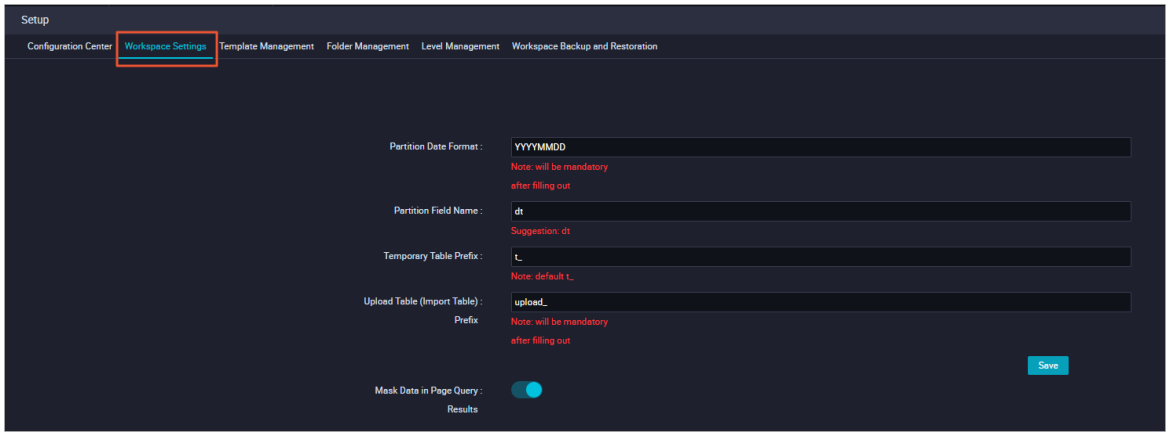
  You can set the DataStudio theme to black or white.

2.5.9.3. Workspace settings

The Workspace Settings tab displays five parameters: Partition Date Format, Partition Field Name, Temporary Table Prefix, Upload Table (Import Table) Prefix, and Mask Data in Page Query Results.

Go to the Workspace Settings tab

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Workspace Settings** tab.



Parameter	Description
Partition Date Format	The default date format of partition field values. You can modify the format as required.
Partition Field Name	The default name of a partition field.
Temporary Table Prefix	The prefix of temporary table names. By default, tables with the prefix t_ in their names are identified as temporary tables.
Upload Table (Import Table) Prefix	The prefix of the names of tables uploaded on the **DataStudio** page.
Mask Data in Page Query Results	Specifies whether to de-identify data in the query results. When the switch is turned on, the result returned for an ad hoc query node in the current workspace will be de-identified.

Enable de-identification for DataWorks workspaces

Data de-identification for DataWorks needs to be enabled in workspaces one by one. After data de-identification is enabled, the result returned for an ad hoc query node in the current workspace will be de-identified. The underlying storage data is not affected because only dynamic de-identification is performed.

> **Note** For example, data de-identification is enabled in workspace A but not workspace B. If you initiate a request in workspace B to query tables in workspace A, the query result is displayed in plaintext.

On the **Workspace Settings** tab, turn on **Mask Data in Page Query Results**. After you click **Save**, the result returned for an ad hoc query node in the current workspace will be de-identified.

> **Note** By default, the Mask Data in Page Query Results switch is turned off and you are not allowed to download de-identified data.

After data de-identification is enabled for DataWorks workspaces, the data types listed in the following table are de-identified by default.

Type	Data de-identification rule	Raw data	De-identified data
ID card number	Only the first and last digits in a 15-digit or an 18-digit ID card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	512345678943215678	5**********8
Mobile number	Only the first three and last two digits in a mobile number in mainland China are displayed in plaintext. All the other digits are displayed as asterisks (*).	18112345678	181******78
Email address	If the string before the at sign (@) in an email address contains three or more characters, only the leftmost three characters are displayed in plaintext, followed by three asterisks (*). If the string before the at sign (@) contains only one or two characters, the entire string is displayed in plaintext, followed by three asterisks (*).	• eftry.abc@gmail.com • af@abc.com	• eft***@gmail.com • af***@abc.com
Bank card number	Only the last four digits in a credit card number or deposit card number are displayed in plaintext. All the other digits are displayed as asterisks (*).	• 1234576834509782 • 643257829145430986	• **********9782 • **********0986
IP address or MAC address	Only the first segment in an IP address or a MAC address is displayed in plaintext. All the other characters are displayed as asterisks (*).	• 192.000.0.0 • ab:cd:11:a3:a0:50	• 192.***. *. * • ab:**:**:**:**:**

Type	Data de-identification rule	Raw data	De-identified data
License plate number	Only the one-character provincial abbreviation and the last three characters in a license plate number in mainland China are displayed in plaintext. All the other characters are displayed as asterisks (*).	• (One-character provincial abbreviation)AP555B • (One-character provincial abbreviation)ADP555T	• (One-character provincial abbreviation)A**55B • (One-character provincial abbreviation)A***55T

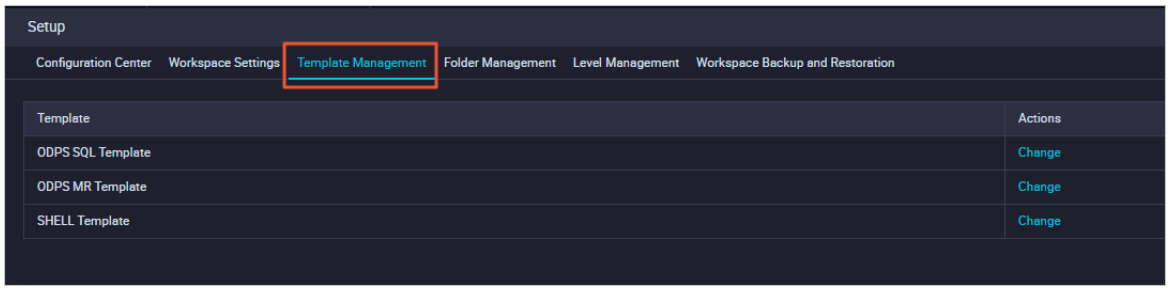
Note If you want to de-identify more data types or have special requirements on the de-identified data formats, complete your de-identification settings in Data Security Guard. The feature of de-identifying data for DataWorks workspaces must work with Data Security Guard. For more information, see Data security guard.

2.5.9.4. Template management

The Template Management page displays code templates. Workspace administrators can change the display formats of the templates as required.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Template Management** tab.



Note Templates are only available for ODPS SQL, ODPS MR, and Shell nodes.

4. Find the target template and click **Change** in the Actions column.
5. In the **Node Template** dialog box, enter the template as required.
6. Click **Save**.

2.5.9.5. Folder management

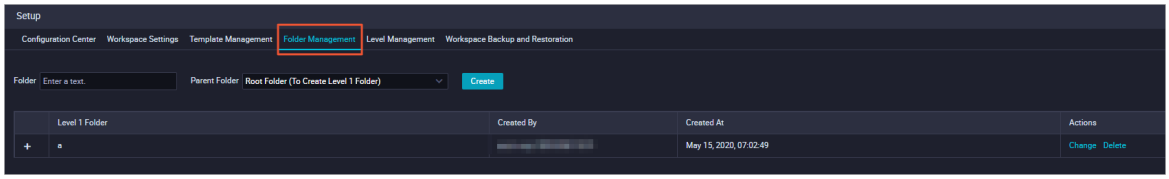
Each workspace can hold a great number of tables. For easy management, you can organize tables in two levels of folders.

Context

Folders are used to store tables. A workspace administrator can add multiple folders and classify tables by purpose and name.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Folder Management** tab.



On the page that appears, you can add, modify, and delete folders.

- Add a folder

  Enter a custom folder name in the **Folder** field, select a parent folder from the **Parent Folder** drop-down list, and then click **Create**.

- Modify a folder

  Find the target folder and click **Change** in the Actions column. In the **Change Folder** dialog box, enter a new folder name and click **OK**.

- Delete a folder

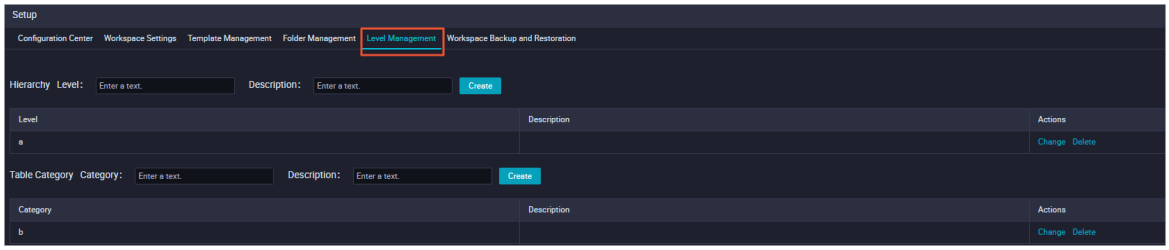
  Find the target folder and click **Delete** in the Actions column. In the **Delete Folder** message, click **OK**.

2.5.9.6. Level management

On the Level Management tab, you can design physical levels of tables.

Procedure

1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click in the lower-left corner.
3. On the **Setup** page, click the **Level Management** tab.



You can classify tables based on their importance. Level management allows you to precisely locate incorrectly organized tables and ensures normal running of published jobs.

If a workspace does not contain default table levels, the workspace owner or workspace administrator must add them as required.

On the **Level Management** tab, you can add, modify, and delete table levels.

- To add a table level, perform the following steps:
  a. In the **Hierarchy** section, set **Level** and **Description**.
  b. Click **Create**.
- To modify a table level, perform the following steps:
  a. Find the target table level and click **Change** in the Actions column.
  b. In the **Change Level** dialog box, modify **Level** and **Description** as needed.
  c. Click **OK**.
- To delete a table level, perform the following steps:
  a. Find the target table level and click **Delete** in the Actions column.
  b. In the **Delete Level** message, click **OK**.

On the **Level Management** tab, you can also add, modify, and delete table categories.

- To add a table category, perform the following steps:
  a. In the **Table Category** section, set **Category** and **Description**.
  b. Click **Create**.
- To modify a table category, perform the following steps:
  a. Find the target table category and click **Change** in the Actions column.
  b. In the **Change Category** dialog box, modify **Category** and **Description** as needed.
  c. Click **OK**.
- To delete a table category, perform the following steps:
  a. Find the target table category and click **Delete** in the Actions column.
  b. In the **Delete Category** message, click **OK**.

2.5.9.7. Workspace backup and restore

On the Workspace Backup and Restoration tab, you can migrate code between workspaces. This topic describes how to back up and restore a workspace.

Prerequisites

Workspaces are created. For more information, see Create a workspace.

Go to the Workspace Backup and Restoration tab

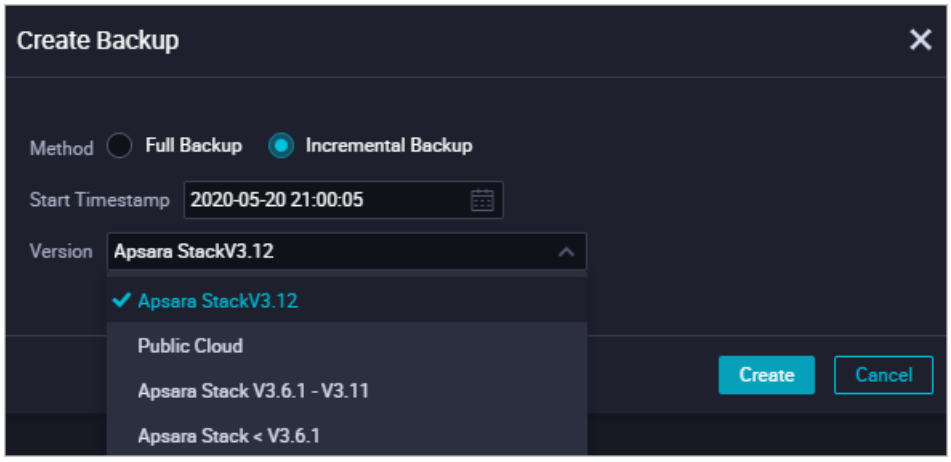
1. Log on to the DataWorks console.
2. On the **Data Analytics** tab, click  in the lower-left corner.
3. On the **Setup** page, click the **Workspace Backup and Restoration** tab.On the page that appears, you can back up and restore workspaces.
   - On the Backup tab, you can compress the node code, node dependencies, resources, and functions in a workspace into one package.
   - On the Restore tab, you can restore a workspace to its original scheduling settings. After the workspace is restored, all nodes in the workspace are saved but not committed.

Back up a workspace

A workspace backup is a compressed package containing the node code, node dependencies, resources, and functions in the workspace.

- Only workspace administrators can export backups and restore data from backups.
- Workflows and node groups of earlier versions cannot be backed up. We recommend that you use the latest version for data analytics.
- A node backed up to a path in the workspace will override the original node with the same name in the path. We recommend that you create another workspace to restore data.
- Data in tables is not backed up when you back up a workspace. You can synchronize the table data in the following ways:
  - Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** and configure a MaxCompute connection. Then, create a sync node to back up the data.
  - In workspace A, run the data definition language (DDL) statement `create table select * from workspace B. Table name` to migrate data.

1. On the **Workspace Backup and Restoration** tab, click **Backup**.
2. Click **Create Backup** in the upper-right corner.
3. In the **Create Backup** dialog box, set the parameters as required.



Parameter	Description
Method	The method used to back up the workspace. Valid values: ○ **Full Backup**: Back up all the node code, node dependencies, resources, and functions in the workspace. ○ **Incremental Backup**: Back up all the new or modified nodes from the timestamp specified by the **Start Timestamp** parameter to the current time. **Note** If you use the incremental backup method, make sure that the dependencies between incremental sync nodes are correct. Otherwise, the workspace may fail to be restored. We recommend that you set this parameter to **Full Backup**.

Parameter	Description
Start Timestamp	The start time point at which data in the workspace is backed up. This parameter is available only when you set **Method** to **Incremental Backup**.
Version	The version of the workspace to be backed up. Valid values: **Apsara StackV3.12**, **Public Cloud**, **Apsara Stack V3.6.1 - V3.11**, and **Apsara Stack < V3.6.1**.

4. Click **Create**.After the data is backed up, click **Download** to download the backup data to a local device.

Restore a workspace

1. On the **Workspace Backup and Restoration** tab, click **Restore**.
2. Click **Restore** in the upper-right corner.
3. In the **Restore** dialog box, click **Select File**.

> **Note** You can upload the compressed package that you previously backed up to the workspace.

4. Click **Restore**.
5. Click **Set Compute Engine Mapping**. In the dialog box that appears, set the mapping between the compute engines of the current workspace and the destination workspace.
6. Click **OK**.If the workspace that you backed up contains multiple compute engines, the system scans all compute engine instances during restoration. The system only restores nodes of the existing compute engines in the workspace to be restored. In this case, you must configure the mappings between the compute engines before restoring the destination workspace.

> **Note**
>
> - If the workspace to be restored does not contain a compute engine type such as E-MapReduce, or no instance is available for the compute engine type, nodes of this engine type are not restored.
> - Compute engine mappings must be configured for custom node types.

2.5.10. Deploy

2.5.10.1. Deploy nodes

In a rigorous data development process, developers develop and debug code and configure dependencies and scheduling properties for nodes in the development environment. Then, developers commit and deploy the nodes to run them in the production environment.

DataWorks workspaces in standard mode can process data seamlessly from the development environment to the production environment within a single workspace. We recommend that you use workspaces in standard mode to develop and produce data.

Deploy nodes in a workspace in standard mode

Each DataWorks workspace in standard mode is linked with two MaxCompute projects, one as the development environment and the other as the production environment. You can directly commit and deploy nodes from the development environment to the production environment.

Follow these steps:

1. On the **DataStudio** page, configure and debug the code of nodes. Then, double-click the target workflow in the left-side navigation pane. On the dashboard of the workflow that appears, click the **Submit** icon to check whether the dependencies between nodes are correct and commit the nodes.
2. After the nodes are committed, click the **Deploy** icon.
3. On the **Create Deploy Task** page that appears, select the target nodes and click **Add to List**. The nodes are added to the to-be-deployed node list.

   You can search for nodes by condition, such as the committer, node type, change type, time when a node is committed, node name, and node ID. If you click Deploy Selected, the selected nodes are deployed to the production environment.
4. Click **View List**. In the Nodes to Deploy dialog box that appears, click **Deploy All**. All nodes in the list are deployed to the production environment.

> **Note** Workspaces in standard mode protect tables in the production environment from being manipulated, and therefore provide the stable, secure, and reliable production environment. We recommend that you use workspaces in standard mode to deploy and run nodes.

Clone nodes between workspaces in basic mode

You cannot deploy nodes in workspaces in basic mode. If you want to isolate the development environment from the production environment for workspaces in basic mode, create two workspaces, one for development and the other for production. You can clone nodes from the development workspace to the production workspace.

As shown in the following figure, two workspaces in basic mode are created, one for development and the other for production. You can use the cross-workspace cloning feature to clone nodes from Workspace A to Workspace B, and then commit the cloned nodes to the scheduler for scheduling.

> **Note**
>
> - Permission requirement: Only workspace administrators and Resource Access Management (RAM) users who have the O&M permissions can clone nodes.
> - Workspace type: You can only clone nodes in workspaces in basic mode, but cannot clone those in workspaces in standard mode.
> - Prerequisites: The source workspace in basic mode and the destination workspace in basic mode are created.

1. Commit nodes.

   After you create and configure nodes in the source workspace, commit the nodes on the dashboard of the target workflow.

2. Click **Cross-Workspace Cloning**.
3. On the Create Clone Task page that appears, select the target nodes and the destination workspace, and then click **Add to List**.
4. Clone the nodes. Click **View List**. In the To-Be-Cloned Nodes dialog box that appears, check the nodes to be cloned and click **Clone All**.

   In the Create Clone Task dialog box that appears, click **Clone**.
5. View the cloned nodes.

   You can view the successfully cloned nodes on the View Clone Tasks page of the source workspace.

   Switch to the destination workspace. You can find that the nodes are cloned from the source workspace.

2.5.10.2. Overview of cross-workspace cloning

For workspaces under the same Apsara Stack tenant account, you can use the cross-workspace cloning feature to clone and deploy workflows across these workspaces. You can also use this feature to clone nodes, such as computing or sync nodes, across workspaces. This topic describes how to process the dependencies between nodes during cross-workspace cloning.

If you clone nodes across workspaces by using the **cross-workspace cloning** feature, DataWorks automatically changes the output names of the cloned nodes in the destination workspace to distinguish nodes in different workspaces under the same Apsara Stack tenant account. This allows you to successfully clone node dependencies.

> **Note** Cross-workspace cloning cannot be used to clone nodes across workspaces in different regions.

You can set the owner of cloned nodes in the destination workspace to **Default** or **Clone Task Creator**.

- If you clone nodes owned by the workspace administrator:

  After the nodes are cloned to the destination workspace, their owner is set to the original owner preferentially. If the original owner is not added to the destination workspace, you will become the owner.
- If you clone nodes owned by yourself:

  After the nodes are cloned to the destination workspace, their owner is set to you preferentially. If you are not added to the destination workspace, you are asked whether to change the owner. If you agree to change the owner, you will be added to the destination workspace and become the owner of the cloned nodes. If you do not agree to change the owner, the clone task is canceled.

Clone a workflow

Assume that the output of the task_A node in the project_1 workspace is project_1.task_A_out. If you clone a workflow that contains the task_A node to the destination workspace project_2, the node output name changes to project_2.task_A_out in the destination workspace.

Clone node dependencies

Assume that the task_ B node in the project_1 workspace depends on the task_A node in the project_3 workspace. If you clone the task_B node in the project_1 workspace to the destination workspace project_2, the dependency between the task_A and task_B nodes is also cloned. The task_B node in the project_2 workspace also depends on the task_A node in the project_3 workspace.

2.5.10.3. Clone nodes across workspaces

This topic describes how to clone nodes across workspaces with an example of cloning a workflow from one workspace to another.

Prerequisites

Two workspaces named Weisong_dataworks_test and Weisong_dataworks_test2 respectively are created. For more information about how to create a workspace, see Create a workspace.

Context

You can clone nodes across workspaces in the following scenarios:

- Clone nodes from a workspace in the basic mode to another workspace in the basic mode.
- Clone nodes from a workspace in the basic mode to another workspace in the standard mode.

After you clone a node, the folder and workflow to which the node belongs are cloned to the destination workspace. Any change to the node, folder, or workflow can also be cloned to the destination workspace.

Procedure

1. Log on to the DataWorks console. On the DataStudio page that appears, switch to the Weisong_dataworks_test workspace in the top navigation bar.
2. Select the target workflow. In the Data Analytics section, double-click the target workflow. On the workflow configuration page that appears, click **Cross-Workspace Cloning** in the upper-right corner. The Create Clone Task page appears.
3. Select the destination workspace, node type, and change type. On the **Create Clone Task** page, set **Target Workspace** to Weisong_dataworks_test2. Select the node type and change type of the node for the clone task as required, and select one or more target nodes that appear in the list. Then, click **Clone Selected**.
4. In the **Create Clone Task** dialog box that appears, check the destination workspace, target node, and change type, and then click **Clone**.
5. After the system message that indicates the target node is cloned successfully and is being committed to the destination workspace appears, switch to the Weisong_dataworks_test2 workspace. In the **Data Analytics** section, view the workflow that has been successfully cloned to the current workspace.

2.5.11. Create an ad hoc query node

The Ad-Hoc Query tab allows you to test your code in the development environment. You can check for errors and check whether your code works as expected.

Context

You do not need to commit and deploy ad hoc query nodes or configure scheduling policies for ad hoc query nodes. You can configure scheduling policies only for nodes created under **Business Flow** on the **Data Analytics** tab.

Create a folder

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Ad-Hoc Query** icon.Click in the lower-left corner to show or hide the left-side navigation pane.
3. On the Ad-Hoc Query tab, move the pointer over and select **Folder**.
4. In the **Create Folder** dialog box, set **Folder Name** and **Location**.

   > **Note**
   > - The folder name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
   > - DataWorks supports multi-level folders. You can save a newly created folder under another folder that already exists.

5. Click **Commit**.

Create an ad hoc query node

You can create **ODPS SQL** and **Shell** nodes on the **Ad-Hoc Query** tab. This topic describes how to create an ODPS SQL node.

1. On the **Ad-Hoc Query** tab, right-click the target folder and choose **Create Node > ODPS SQL**.
2. In the **Create Node** dialog box, set **Node Name** and **Location**.

   > **Note** The node name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.

3. Click **Commit**.
4. On the node configuration tab that appears, enter an SQL statement.
5. Click in the toolbar.

2.5.12. View runtime logs

The Runtime Logs tab displays the records of all nodes that have been run in the last three days. You can click a node to view its runtime logs.

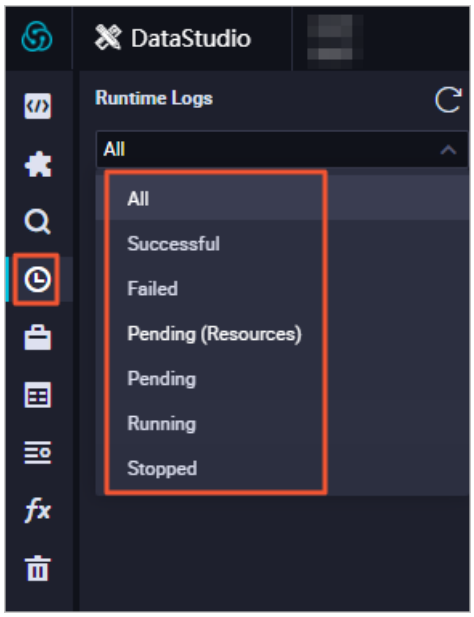
Context

The runtime logs are retained for only three days.

Procedure

1. Log on to the DataWorks console.

2. On the left-side navigation submenu, click the **Runtime Logs** icon. Click in the lower-left corner to show or hide the left-side navigation pane.
3. Select a node state from the drop-down list to view the runtime logs of nodes in the specified state.



4. Click a record to view the runtime log on the right. If you need to save the SQL statements in the runtime log, click in the toolbar. In the **Create Node** dialog box, set the parameters and click **Commit** to save the SQL statements that have been run as an ad hoc query node.

2.5.13. View tenant tables

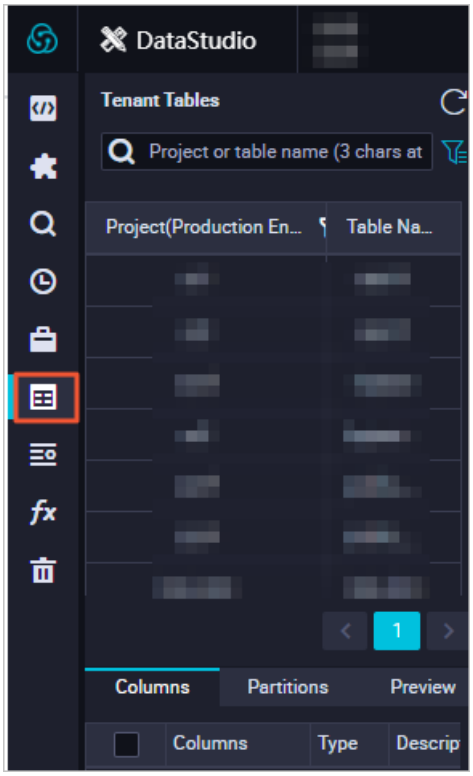
On the Tenant Tables tab, you can view tables of all workspaces of the current tenant account.

Prerequisites

The **Tenant Tables** tab appears only after you bind a MaxCompute compute engine on the **Project Management** page. For more information, see Configure a workspace.

Procedure

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Tenant Tables** icon.Click in the lower-left corner to show or hide the left-side navigation pane.
3. View MaxCompute tenant tables.



Parameter or tab	Description
Project Name	The name of the workspace in the corresponding environment. Click next to the search box and select the target environment to switch to the environment. **Note** ◦ For a workspace in standard mode, the **Tenant Tables** tab displays tables in both the development environment and the production environment. For a workspace in basic mode, the **Tenant Tables** tab displays only the tables in the production environment. ◦ The current environment is marked in blue.
Table Name	The name of the table in the corresponding workspace.
Columns tab	Displays the name, data type, and description of fields in the table.

Parameter or tab	Description
Partitions tab	Displays the partition information of the current table. A maximum of 60,000 partitions are supported. If you have specified the TTL for partitions, the number of partitions depends on the TTL. Notice The partition information is displayed only for MaxCompute tenant tables.
Preview tab	Displays the data of the current table. Notice You can preview only the data of MaxCompute tenant tables.

2.5.14. Manage tables

This topic describes how to view, modify, and delete MaxCompute tables, and the basic knowledge about data hierarchy.

Prerequisites

The **Workspace Tables** tab appears only after you bind a MaxCompute compute engine on the **Project Management** page. For more information, see Configure a workspace.

Manage tables

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Workspace Tables** icon.Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. View and manage tables.The following section describes how to view, modify, and delete a MaxCompute table. For more information about how to create a table, see Create a MaxCompute table.

Operation	Description

Operation	Description
View a table	Click  next to the search box and select the target environment to switch to the environment. **Note** ◦ For a workspace in standard mode, the **Workspace Tables** tab displays tables in both the development environment and the production environment. For a workspace in basic mode, the **Workspace Tables** tab displays only the tables in the production environment. ◦ The current environment is marked in blue. Double-click a table to view its details on the table configuration tab.
Import data to a table	On the **Workspace Tables** tab, click  to import data to a table. For more information, see Create tables and import data.

Divide a data warehouse into layers

In the **Physical Model** section of the configuration tab of a table, you can define table layers for a data warehouse. This allows you to have better planning and control over your data.

Typically, a data warehouse consists of the following layers:

- ODS: The ODS layer stores raw data of the source system based on the original data structure. The ODS layer serves as the data staging area of the data warehouse. It imports basic data to MaxCompute and records historical changes of basic data.
- CDM: The CDM layer consists of the dimension data (DIM), data warehouse detail (DWD), and data warehouse service (DWS) layers. The CDM layer processes and integrates the data of the ODS layer to define conformed dimensions, create reusable detailed fact tables for data analysis and statistics collection, and aggregate common metrics.
  - The DIM layer defines conformed dimensions for an enterprise based on the concepts of dimensional modeling. It reduces the risk of inconsistent statistical criteria and algorithms.

    Tables at the DIM layer are also called logical dimension tables. Generally, each dimension corresponds to a logical dimension table.
  - The DWS layer is driven by analyzed subjects during data modeling. Based on the metric requirements of upper-layer applications and products, the DWS layer creates fact tables to aggregate common metrics and builds a physical data model by using wide tables. The DWS layer creates statistical metrics in compliance with uniform naming conventions and statistical criteria, provides common metrics for the upper layer, and generates aggregate wide tables and detailed fact tables.

    Tables at the DWS layer are also called logical aggregate tables, which are used to store derived metrics.

- The DWD layer is driven by business processes during data modeling. It creates detailed fact tables at the finest granularity based on each specific business process. In combination with the data usage habits of an enterprise, you can duplicate some key attribute fields of dimensions in detailed fact tables to create wide tables.

  Tables at the DWD layer are also called logical fact tables.

- ADS: The ADS layer stores personalized statistical metrics of data products. It processes the data of the CDM and ODS layers.

2.5.15. View built-in functions

The Built-In Functions tab displays functions built in MaxCompute. You can view the types, description, and examples of functions on this tab.

Procedure

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Built-In Functions** icon.Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. View the types, description, and examples of the built-in functions.Functions are categorized into aggregate functions, analytic functions, date functions, mathematical functions, string functions, and other functions. The preceding functions are built in MaxCompute. You can click a function to view its description.

2.5.16. Manage deleted nodes

DataWorks provides a recycle bin to store all deleted nodes in the current workspace. You can restore or permanently delete the nodes.

Go to the Recycle Bin tab

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Recycle Bin** icon. Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. View all the deleted nodes in the current workspace. On this tab, you can delete or restore a deleted node.

   > **Notice**
   > - The recycle bin displays only 100 nodes. If more than 100 nodes are deleted, the nodes deleted earlier are deleted permanently from the recycle bin.
   > - Deleted node groups are not displayed in the recycle bin.

Restore a node in the recycle bin

1. On the **Recycle Bin** tab, right-click a deleted node.
2. Select **Restore**.
3. In the **Restore Node** message, click **OK**.

> **Note** After you restore a node, a new node ID is generated for scheduling and all the information about the node is restored.

Permanently delete a node from the recycle bin

1. On the **Recycle Bin** tab, right-click a deleted node.
2. Select **Delete**.
3. In the **Delete** message, click **OK**.

> **Note** Nodes permanently deleted from the recycle bin cannot be restored. The recycle bin displays only deleted nodes.

2.5.17. Create a manually triggered workflow

In a manually triggered workflow, all nodes must be manually triggered, and cannot be automatically scheduled by DataWorks. Therefore, you do not need to specify parent nodes or outputs for nodes in manually triggered workflows.

Create a manually triggered workflow

1. Log on to the DataWorks console.
2. On the left-side navigation submenu, click the **Manually Triggered Workflows** icon.

   Click ☰ in the lower-left corner to show or hide the left-side navigation pane.
3. Right-click **Manually Triggered Workflows** and select **Create Workflow**.
4. In the **Create Workflow** dialog box, set **Workflow Name** and **Description**.

   > **Notice** The workflow name must be 1 to 128 characters in length and can contain letters, digits, underscores (_), and periods (.). It is not case-sensitive.
5. Click **Create**.

Composition of a manually triggered workflow

> **Note** We recommend that you create a maximum of 100 nodes in a manually triggered workflow.

A manually triggered workflow consists of the nodes of the following modules. After you create a manually triggered workflow, open this workflow and create nodes of various types for each module. For more information, see Node types.

- **Data Integration**

  Double-click **Data Integration** under the created workflow to view all the data integration nodes.

  Right-click **Data Integration** and choose **Create > Batch Synchronization** to create a batch sync node. For more information, see Create a batch sync node.

- **MaxCompute**

  The MaxCompute compute engine consists of data analytics nodes, such as ODPS SQL, SQL Snippet, ODPS Spark, PyODPS, ODPS Script, and ODPS MR nodes. You can also view and create tables, resources, and functions.

  - **Data Analytics**

    Show **MaxCompute** under the created workflow and right-click **Data Analytics** to create a data analytics node. For more information, see Create an ODPS SQL node, Create an SQL Snippet node, Create an ODPS Spark node, Create a PyODPS node, Create an ODPS Script node, and Create an ODPS MR node.

  - **Table**

    Show **MaxCompute** under the created workflow and right-click **Table** to create a table. You can also view all the tables created for the current MaxCompute compute engine. For more information, see Create a MaxCompute table.

  - **Resource**

    Show **MaxCompute** under the created workflow and right-click **Resource** to create a resource. You can also view all the resources created for the current MaxCompute compute engine. For more information, see Create, reference, and download resources.

  - **Function**

    Show **MaxCompute** under the created workflow and right-click **Function** to create a function. You can also view all the functions created for the current MaxCompute compute engine. For more information, see Register a UDF.

- **Algorithm**

  Click the created workflow and right-click **Algorithm** to create an algorithm. You can also view all the PAI nodes created in the current manually triggered workflow. For more information, see Create a PAI node.

- **General**

  Click the created workflow and right-click **General** to create relevant nodes. For more information, see Create a Shell node and Create a zero load node.

- **UserDefined**

  Click the created workflow and right-click **UserDefined** to create relevant nodes. For more information, see Create a Hologres development node.

GUI elements

The following table describes the icons and tabs on the Manually Triggered Workflows page.

No.	Icon or tab	Description
1	Submit icon	Commits all nodes in the current manually triggered workflow.
2	Run icon	Runs all nodes in the current manually triggered workflow. Nodes in this workflow do not have dependencies, and therefore they can run at a time.
3	Stop icon	Stops all running nodes in the current manually triggered workflow.
4	Deploy icon	Navigates to the Deploy page. On this page, you can deploy some or all nodes that are committed but not deployed to the production environment. Note This icon is available only when the workspace is in standard mode.
5	Go to Operation Center icon	Navigates to the Operation Center page.
6	Box. icon	Box-selects a node group consisting of required nodes.
7	Refresh icon	Refreshes the page of the current manually triggered workflow.
8	Auto Layout icon	Sorts the nodes in the current manually triggered workflow.
9	Zoom In icon	Zooms in the current page.
10	Zoom Out icon	Zooms out the current page.
11	Search icon	Searches for a node in the current manually triggered workflow.

No.	Icon or tab	Description
12	Toggle Full Screen View icon	Displays nodes in the current manually triggered workflow in the full screen.
13	Show Engine Information/Hide Engine Information icon	Shows or hides engine information.
14	Workflow Parameters tab	Allows you to set parameters. Parameters set on this tab have a higher priority than those specified on the corresponding node configuration tab. If two values are set separately, the value set on the Workflow Parameters tab takes effect.
15	Change History tab	Allows you to view the operation records of all nodes in the current manually triggered workflow.
16	Versions tab	Allows you to view the deployment records of all nodes in the current manually triggered workflow.

2.5.18. Editor keyboard shortcuts

This section describes keyboard shortcuts available for the code editor.

Google Chrome in Windows OS

`Ctrl + S` : Save changes to a node.

`Ctrl + Z` : Undo an action.

`Ctrl + Y` : Redo an action.

`Ctrl + D` : Select occurrences.

`Ctrl + X` : Cut a line.

`Ctrl + Shift + K` : Delete a line.

`Ctrl + C` : Copy a line.

`Ctrl + I` : Select a line.

`Alt + Shift + Drag` : Select a block.

`Alt + Click` : Insert an additional cursor.

`Ctrl + Shift + L` : Select all occurrences.

`Ctrl + F` : Search for text in a node.

`Ctrl + H` : Replace text in a node.

`Ctrl + G` : Locate a line.

`Alt + Enter` : Select all matched strings.

`Alt + Up or down arrow` : Move a line up or down.

`Alt + Shift + Up or down arrow` : Duplicate a line.

`Ctrl + Shift + K` : Delete a line.

`Ctrl + Enter or Ctrl + Shift + Enter` : Insert a line break downwards or upwards.

`Ctrl + Shift + Back slash (\)` : Jump to the parenthesis, bracket, or brace that matches the adjacent one.

`Ctrl + Left bracket (]) or right bracket ([)` : Increase or decrease the indent of a line.

`Home or End` : Move the cursor to the beginning or end of a line.

`Ctrl + Home or End` : Move the cursor to the top or bottom of a node.

`Ctrl + Left or Right arrow` : Move the cursor one word to left or right.

`Ctrl + Shift + Left bracket (]) or right bracket ([)` : Hide or show a block.

`Ctrl + K + Left bracket (]) or right bracket ([)` : Hide or show sub-blocks in a block.

`Ctrl + K + 0 or J` : Hide or show all blocks.

`Ctrl + Slash (/)` : Comment out or uncomment the selected lines or blocks.

Google Chrome in Mac OS

`Command-S` : Save changes to a node.

`Command-Z` : Undo an action.

`Command-Y` : Redo an action.

`Command-D` : Select occurrences.

`Command-X` : Cut a line.

`Shift-Command-K` : Delete a line.

`Command-C` : Copy a line.

`Command-I` : Select a line.

`Command-F` : Search for text in a node.

`Option-Command-F` : Replace text in a node.

`Option-Up or down arrow` : Move a line up or down.

`Option-Shift-Up or down arrow` : Duplicate a line.

`Shift-Command-K` : Delete a line.

`Command-Enter or Shift-Command-Enter` : Insert a line break downwards or upwards.

`Shift-Command-Back slash (\)` : Jump to the parenthesis, bracket, or brace that matches the adjacent one.

`Command-Left bracket (]) or right bracket ([)` : Increase or decrease the indent of a line.

`Command-Left or right arrow` : Move the cursor to the beginning or end of a line.

`Command-Up or down arrow` : Move the cursor to the top or bottom of a node.

`Option-Left or right arrow` : Move the cursor one word to left or right.

`Option-Command-Left bracket (]) or right bracket ([)` : Hide or show a block.

`Command-K-Left bracket (]) or right bracket ([)` : Hide or show sub-blocks in a block.

`Command-K-0 or J` : Hide or show all blocks.

`Command-Slash (/)` : Comment out or uncomment the selected lines or blocks.

Insert multiple cursors and select multiple occurrences or lines

`Option-Click` : Insert an additional cursor.

`Option-Command-Up or down arrow` : Insert an additional cursor to the previous or next line.

`Command-U` : Undo a cursor-related operation.

`Option-Shift-I` : Insert a cursor at the end of each selected line.

`Command-G or Shift-Command-G` : Select the next or previous matched string.

`Command-F2` : Select the nearest character of each cursor.

`Shift-Command-L` : Select the nearest word of each cursor.

`Option-Enter` : Select all the matched strings.

`Option-Shift-Drag` : Multi-select lines

`Option-Shift-Command-Up or down arrow` : Extend a selection one line up or down.

`Option-Shift-Command-Left or right arrow` : Extend a selection one character to the left or right.

2.5.19. Use E-MapReduce in DataWorks

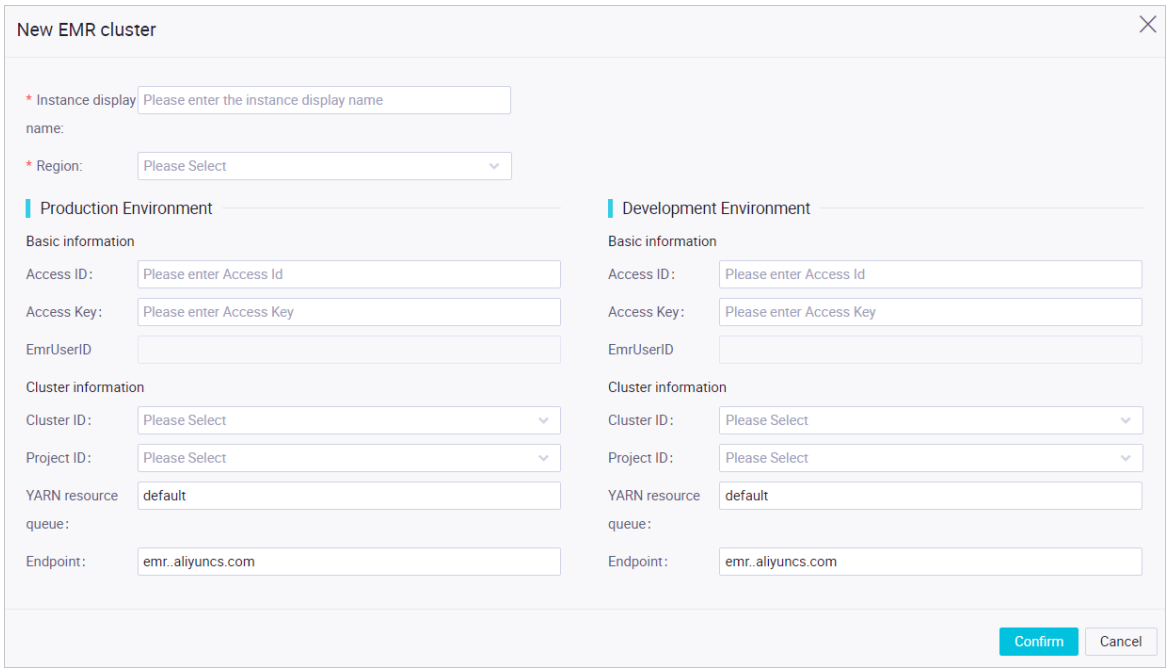
This topic describes how to use E-MapReduce in DataWorks.

Bind an E-MapReduce project to a DataWorks workspace

> **Note** Before you bind an E-MapReduce project to a DataWorks workspace, you must obtain the information about the E-MapReduce project.

1. Log on to the DataWorks console.
2. On the DataStudio page, click  in the upper-right corner. The **Project Management** page appears.

3. Click the **E-MapReduce** tab in the **Computing Engine Information** section. On this tab, you can view the information about all available E-MapReduce compute engines in the current workspace.
4. Click **Add instances**.
5. In the **New EMR cluster** dialog box, set the parameters as required.



Parameter	Description
Instance display name	The name of the E-MapReduce compute engine instance.
Region	The region of the workspace.
Access ID	The AccessKey ID of the account authorized to access the E-MapReduce cluster.
Access Key	The AccessKey secret of the account authorized to access the E-MapReduce cluster.
EmrClusterID	The ID of the E-MapReduce cluster.
Cluster ID	The ID of the user who created the E-MapReduce cluster.
Project ID	The ID of the project in the E-MapReduce cluster.
YARN resource queue	The name of the resource queue in the E-MapReduce cluster. Unless otherwise specified, set the value to *default*.
Endpoint	The endpoint of the E-MapReduce cluster. You can obtain the endpoint in the Apsara Stack Operations console. For more information, see Obtain an endpoint.

6. Click **Confirm**. After the E-MapReduce cluster is bound to your workspace, you can create E-MapReduce nodes on the **DataStudio** page.

> Note If the binding fails, check whether the failure is caused by one of the following reasons:
>
> - The E-MapReduce user ID is bound to another tenant account.
> - The specified cluster name already exists.

Create an E-MapReduce node

E-MapReduce nodes are categorized into four types: EMR Hive, EMR Spark SQL, EMR Spark, and EMR MR. For more information, see Create an EMR MR node.

Reference resource files

E-MapReduce resource files are categorized into two resource types: EMR JAR and EMR File.

Reference E-MapReduce resource files by using the following methods:

- For EMR Hive and EMR MR nodes, add `--@resource_reference{"Resource name"}` at the first line of the code.
- For EMR Spark nodes, add `##@resource_reference{"Resource name"}` at the first line of the code.

Manage data

DataWorks allows you to query E-MapReduce metadata and synchronize the data for data development.

2.6. HoloStudio

2.6.1. Overview

This topic describes what is HoloStudio and the features of HoloStudio.

HoloStudio, which is developed based on Hologres, is an all-in-one online analytical processing (OLAP) development platform deeply integrated with DataWorks. HoloStudio provides Hologres users with standardized and easy-to-use development services and one-stop real-time data warehouse building services by using a visualized and wizard-based user interface. In addition to standard management available in PostgreSQL, HoloStudio provides more interactive analytics features. HoloStudio supports the following features:

- Table management

  HoloStudio allows you to create a PostgreSQL table on a visualized user interface or by using SQL statements. HoloStudio also allows you to create a foreign table sourced from MaxCompute by synchronize the schema with one click and preview and analyze MaxCompute data.

- SQL Console

  The SQL Console module of HoloStudio allows you to use an SQL editor to obtain query results in seconds.

- Data analytics

Based on the underlying capabilities of DataWorks, HoloStudio allows you to create multiple foreign tables at a time and synchronize data with one click and provides you with the one-stop, stable, and efficient extract-transform-load (ETL) service.

- Seamless connection to the Hologres console

  HoloStudio is seamlessly connected to the Hologres console. In the Hologres console, you can manage Hologres instances, users, and databases on a visualized user interface.

2.6.2. Bind a Hologres database to the current workspace

This topic describes how to bind a Hologres database to the current workspace.

If you need to use HoloStudio for data development, bind a Hologres database to the current workspace. Perform the following steps:

1. Log on to HoloStudio.

   On the left-side navigation submenu, click **PG management**. On the PG management tab, move the pointer over the Create icon and select **Database**.

2. In the Create Database dialog box, set relevant parameters.

   In the **Create Database** dialog box, set relevant parameters and click **Test connectivity**. If the system message indicating that the connectivity test is passed appears, the specified database is connected. Then, click **Complete**.

Parameter	Description	Remarks
Connect To	The type of the data store.	The value is automatically generated and cannot be changed.
Server	The endpoint of the Hologres instance.	You can view the endpoint on the Basic Information page of the Hologres instance in the Hologres console.
Port	The port number of the Hologres instance.	You can view the port number on the Basic Information page of the Hologres instance in the Hologres console.
Database Name	The name of the Hologres database to be bound to the current workspace.	In actual scenarios, create a Hologres database and set this parameter to the name of the database to bind the database to the current workspace.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey ID.

Parameter	Description	Remarks
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey secret.
JDBC Extension	The extension parameters used to establish a Java Database Connectivity (JDBC) connection to Hologres.	Example: `?preferQueryMode=simple&tcpKeepAlive=true`
Test connectivity	Tests whether the database is connected.	N/A

3. Use the Hologres database in HoloStudio.

   After the Hologres database is bound to the current workspace, click the Refresh icon on the **PG management** tab. After the database appears, you can use the database in HoloStudio.

2.6.3. SQL Console

The SQL Console in HoloStudio is an editor for executing SQL statements. In the SQL Console, you can execute SQL statements to analyze data in Hologres and obtain the query results. This topic describes the basic features and usage of the SQL Console in HoloStudio.

Folder

The Folder module stores new ad hoc queries, which helps you manage ad hoc queries.

In the left-side navigation pane, click **SQL Console**. Move the pointer over the Create icon and select **Folder**. In the Create Folder dialog box, enter a folder name and click Commit to create a folder. You can create an ad hoc query in the folder and execute standard SQL statements on tables. You can also right-click a table in the folder and select the relevant menu item to move, rename, or delete the table.

SQL Console

The SQL Console module allows you to create ad hoc queries and execute standard SQL statements.

1. Create an ad hoc query.

   In the left-side navigation pane, click **SQL Console**. Move the pointer over the Create icon and select **SQL Console**. In the Create Node dialog box, set relevant parameters.

   The following table describes the parameters for creating an ad hoc query.

Parameter	Description
Node Name	The name of the ad hoc query. The name can contain letters, digits, underscores (_), and periods (`.`).
Location	The folder where the ad hoc query is stored.

Parameter	Description
Database	The target database in which the ad hoc query is run.

2. Run the ad hoc query.

   Write the SQL statements used in the ad hoc query and click the **Run** icon to run the query. Then, you can check the query result. The following example shows how to create a table, import data to the table, and then query the table:

```
CREATE TABLE supplier (
 s_suppkey bigint NOT NULL,
 s_name text NOT NULL,
 s_address text NOT NULL,
 s_nationkey bigint NOT NULL,
 s_phone text NOT NULL,
 s_acctbal bigint NOT NULL,
 s_comment text NOT NULL,
PRIMARY KEY (s_suppkey)
);

INSERT INTO supplier VALUES
(1, 'Supplier#000000001', 'gf0JBoQDd7tgrzrddZ', 17, '27-918-335-1736', 575594, 'each slyly above the careful'),
(6, 'Supplier#000000006', 'tQxuVm7s7CnK', 14, '24-696-997-4969', 136579, 'final accounts. regular dolphins use against the furiously ironic decoys. '),
(10, 'Supplier#000000010', 'Saygah3gYWMp72i PY', 24, '34-852-489-8585', 389191, 'ing waters. regular requests ar'),
(18, 'Supplier#00000001', 'PGGVE5PWAMwKDZw', 16, '26-729-551-1115', 704082, 'accounts snooze slyly furiously bold'),
(39, 'Supplier#000000039', 'SYpEPWr1yAFHaC91qjFcijjeU5eH', 8, '18-851-856-5633 611565', 88990, 'le slyly requests. special packages shall are blithely. slyly unusual packages sleep'),
(48, 'Supplier#000000048', 'FNPMQDuyuKvTnLXXaLf3Wl6OtONA6mQlWJ', 14, '24-722-551-9498',563062, 'xpress instructions affix. fluffily even requests boos');

SELECT * FROM supplier;
```

   The following table describes the GUI elements on the node editing tab.

GUI element	Description
SQL editor	You can write SQL statements in the SQL editor.

GUI element	Description
Save icon	You can click this icon to save all statements in the SQL editor.
Run icon	You can click this icon to execute all statements in the SQL editor. The result appears on the Result tab. You can also select an SQL statement to be executed. In this case, the system only executes this statement.
Refresh icon	You can click this icon to refresh the content in the SQL editor. The system retains only the saved content after the refresh.
Stop icon	You can click this icon to stop executing SQL statements.
Runtime Log	You can check the execution results and, if any, error messages.
Result	You can check the table content after the SQL statements are executed.
Steal Lock icon	You can click this icon to unlock the locked SQL Console.

HoloStudio also allows you to directly manage the queried data. For example, you can hide columns, copy data, and search for data.

Note For statements without results returned, such as the `CREATE TABLE` statement, only operational logs are generated after the statements are executed.

2.6.4. PostgreSQL management

2.6.4.1. Manage databases

The PG management module of HoloStudio allows you to manage databases and tables with one click in a visualized manner. In addition, HoloStudio supports interactive queries with a response time within seconds. This topic describes how to use the PG management module to manage databases.

Create a database

The PG management module allows you to connect databases to HoloStudio and manage the connected databases in a visualized manner. Before you connect a database to HoloStudio, create the database in the Hologres console or by executing SQL statements in the SQL Console. The following example demonstrates how to create a database by executing SQL statements in the SQL Console and connect the database to HoloStudio:

1. Create an ad hoc query.

Log on to the DataWorks console as a superuser and go to the HoloStudio page. In the SQL Console, execute the following SQL statement:

```
create database dbname;
create database testdb; // Create a database named testdb.
```

2. Bind the Hologres database to the current workspace.

   On the HoloStudio page, click **PG management** on the left-side navigation submenu. On the PG management tab, move the pointer over the Create icon and select **Database**.

3. In the Create Database dialog box, set relevant parameters.

   In the **Create Database** dialog box, set relevant parameters and click **Test connectivity**. If the system message indicating that the connectivity test is passed appears, the specified database is connected. Then, click **Complete**.

Parameter	Description	Remarks
Connect To	The type of the data store. Default value: Hologres.	The value is automatically generated and cannot be changed.
Server	The endpoint of the Hologres instance.	You can view the endpoint on the Basic Information page of the Hologres instance in the Hologres console.
Port	The port number of the Hologres instance.	You can view the port number on the Basic Information page of the Hologres instance in the Hologres console.
Database Name	The name of the database to be bound to the current workspace.	The value must be the same as the database name in the CREATE DATABASE statement, for example, **testdb**.
User name	The AccessKey ID of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey ID.
Password	The AccessKey secret of your Apsara Stack tenant account.	You can click the username in the upper-right corner to view the AccessKey secret.
JDBC Extension	The extension parameters used to establish a JDBC connection to Hologres.	Example: `?preferQueryMode=simple&tcpKeepAlive=true`
Test connectivity	Tests whether the database is connected.	If the system message indicating that the connectivity test is passed appears, the specified database is connected.

Delete a database

The PG management module of HoloStudio allows you to delete databases. On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the database you want to delete and select **Delete Database**.

In the Delete Database message, click **Ok** to delete the database.

> **Note** Only a superuser of a database or the database owner configured by a superuser can delete the database.

View database details

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the database for which you want to view details and select **Database details**.

2.6.4.2. Manage tables

Similar to PostgreSQL, Hologres manages data by using tables. The PG management module of HoloStudio allows you to manage tables in a visualized manner. You can create, check, or delete a table with one click. This topic describes how to use the PG management module of HoloStudio to manage tables.

Create a table

1. Create a table.

   On the homepage of HoloStudio, click **Create Table**. Alternatively, click **PG management** on the left-side navigation submenu. On the PG management tab, move the pointer over the Create icon and select **Table**.

2. Edit the table content and attributes.

   On the tab that appears, edit the table content and attributes, and click **Commit**. The following figure shows an example of a column-oriented table with primary keys.

Section or tab	Parameter	Description
General	Interactive Analytics Database	The database where the table resides.
	Table Name	The name of the table.
	Description	The description of the table.
	Field Name	The name of the field in the table.
	Data Type	The data type of the field.
	Primary Key Field	Specifies whether to use the field as the primary key for the table.

Section or tab	Parameter	Description
Field	Optional	Specifies whether the field can be null.
	Array	Specifies whether the field is an ordered array of elements.
	Description	The description of the field.
	Actions	The actions that you can perform on the field. For example, you can delete the field from the table, or move up or down the position of the field in the table.
Properties	Storage Mode	The storage mode of the table. Valid values: Row Store and Column Store. Default value: Column Store.
	Lifecycle (Seconds)	The lifecycle of the table. Default value: Permanent.
	Clustered Index	The index used for sorting.
	Dictionary Code Columns	The column based on whose values a dictionary mapping is built.
	Bitmap Column	The column on which bit code is built.
Partitioned Table	PARTITION BY LIST	The partition field.

Check a table

1. View the DDL statement used to create the table.

   On the left-side navigation submenu, click **PG management**. Double-click the table you want to check and click **Generate DDL Statement** to check the SQL statement used to create the table.

2. Preview data.

   On the left-side navigation submenu, click **PG management**. Double-click the table you want to check and click **Data Preview** to check the content of the table. If the table contains no data, you can only view the fields of the table.

Delete a table

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the table you want to delete and select **Delete Table**.

In the Delete message, click **Ok**.

2.6.4.3. Manage foreign tables

In Hologres, a foreign table does not store data but maps the table from the external data source. The PG management module of HoloStudio allows you to create, query, or delete foreign tables. You can only analyze foreign tables sourced from MaxCompute. This helps you obtain the query results.

This topic describes how to use the PG management module of HoloStudio to manage foreign tables.

Create a foreign table

On the left-side navigation submenu, click **PG management**. On the PG management tab, move the pointer over the Create icon and select **External Table**. On the tab that appears, set parameters for creating a foreign table and click **Commit**. An existing MaxCompute table is used in the following example. After you search for a MaxCompute table by entering its name, HoloStudio automatically generates a foreign table based on the fields of the MaxCompute table after you click Commit.

> **Note**
>
> 1. Before you create a foreign table in Hologres, make sure that its source table exists in a MaxCompute project.
> 2. The fields of a foreign table in Hologres have a one-to-one mapping with those of the source table in MaxCompute. You can query specific fields or all fields.

Section or icon	Parameter	Description
General	Interactive Analytics Database	The database where the foreign table to be created resides.
	Table Name	The name of the foreign table.
External Service	Types	The service type of the external table. You can only set this parameter to MaxCompute.
Table	Table	The source table in MaxCompute to be mapped.
Commit	Commit	You can click this button to commit the foreign table that you create.

Check a foreign table

1. **Preview data.**

   On the left-side navigation submenu, click **PG management**. On the PG management tab, double-click the foreign table you want to check, and click **Data Preview** to check the content of the foreign table.

2. **View the DDL statement used to create the table.**

   On the left-side navigation submenu, click **PG management**. On the PG management tab, double-click the foreign table you want to check, and click **Generate DDL Statement** to check the SQL statement used to create the foreign table.

Delete a foreign table

On the left-side navigation submenu, click **PG management**. On the PG management tab, right-click the foreign table you want to delete and select **Delete Table**. In the Delete message, click **Ok** to delete the foreign table.

2.6.5. Data analytics

2.6.5.1. Overview

The Data Analytics module of HoloStudio is seamlessly integrated with DataWorks for node scheduling and provides all-in-one, stable, and efficient extract, transform, load (ETL) services. It can also synchronize MaxCompute table schemas and data and allows you to upload local files for data analytics.

The Data Analytics module consists of the following submodules:

1. Folder: stores data analytics nodes, helping you manage data analytics nodes of each database.
2. Interactive Analytics Development: is integrated with DataWorks to schedule ETL nodes.
3. One-click MaxCompute table structure synchronization: allows you to create multiple foreign tables sourced from MaxCompute at a time.
4. One-click MaxCompute data synchronization: provides a visualized user interface for you to synchronize MaxCompute data to Hologres.
5. One-click local file Upload: allows you to upload local files to Hologres.

Folder

Folders store data analytics nodes, helping you manage data analytics nodes of each database.

On the left-side navigation submenu, click **Data Analytics**. On the Data Analytics tab, move the pointer over the Create icon and select **Folder**. In the Create Folder dialog box, enter a folder name and click **Commit**.

2.6.5.2. Use the Interactive Analytics Development submodule

The Interactive Analytics Development submodule is seamlessly integrated with DataWorks. You can use HoloStudio to import data from MaxCompute to Hologres. You can also use DataWorks to schedule nodes to periodically import data to Hologres. This topic describes how to use HoloStudio to map the source data stored in a MaxCompute table to Hologres for periodic scheduling.

1. **Prepare a MaxCompute table.**

Create a table in MaxCompute and import data to the table. You can also select a table with data from Data Map. In this example, an existing table in Data Map is used. The following Data Definition Language (DDL) statement is used to create the table:

```
CREATE TABLE IF NOT EXISTS bank_data_odps
(
 age           BIGINT COMMENT 'age',
 job           STRING COMMENT 'job type',
 marital       STRING COMMENT 'marital status',
 education     STRING COMMENT 'education level',
 card          STRING COMMENT 'credit card available or not',
 housing       STRING COMMENT 'mortgage',
 loan          STRING COMMENT 'loan',
 contact       STRING COMMENT 'contact',
 month         STRING COMMENT 'month',
 day_of_week   STRING COMMENT 'day in a week',
 duration      STRING COMMENT 'duration',
 campaign      BIGINT COMMENT 'number of contacts during the campaign',
 pdays         DOUBLE COMMENT 'interval from the last contact',
 previous      DOUBLE COMMENT 'number of contacts with the customer',
 poutcome      STRING COMMENT 'result of the previous marketing campaign',
 emp_var_rate  DOUBLE COMMENT 'employment change rate',
 cons_price_idx DOUBLE COMMENT 'consumer price index',
 cons_conf_idx DOUBLE COMMENT 'consumer confidence index',
 euribor3m     DOUBLE COMMENT 'euro deposit rate',
 nr_employed   DOUBLE COMMENT 'number of employees',
 y             BIGINT COMMENT 'fixed time deposit available or not'
);
```

2. **Create a foreign table.**

   Go to the HoloStudio page. On the left-side navigation submenu, click **PG management** or **SQL Console**. On the tab that appears, create a foreign table for mapping data in the MaxCompute source table. In this example, use the following SQL statements to create a foreign table:

```
BEGIN;
CREATE FOREIGN TABLE if not EXISTS bank_data_foreign_holo (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8
)
SERVER odps_server
OPTIONS (project_name 'projectname', table_name 'bank_data_odps');
GRANT SELECT ON bank_data_foreign_holo TO PUBLIC;
COMMIT;
```

Note The OPTIONS parameter contains two fields: project_name, which is the name of the MaxCompute project, and table_name, which is the name of the MaxCompute table.

3. **Create a data storage table.**

   Create a table in HoloStudio to receive and store data. The fields in this table must be of the same data types as those in the foreign table. In this example, use the following SQL statements to create the storage table:

```
BEGIN;
CREATE TABLE if not EXISTS bank_data_holo (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8,
 ds text NOT NULL
)
PARTITION  BY LIST(ds);
CALL SET_TABLE_PROPERTY('bank_data_holo', 'orientation', 'column');
CALL SET_TABLE_PROPERTY('bank_data_holo', 'time_to_live_in_seconds', '700000');
COMMIT;
```

4. **Create a partitioned table.**

   On the HoloStudio page, click **Data Analytics** on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **Interactive Analytics Development** to create a Hologres development node. Then, go to the SQL editor of the node and enter SQL statements to create a partitioned table for obtaining the required data. After you enter the SQL statements, click the **Run** icon. In the Field dialog box, set a value for the ${bizdate} parameter. After the SQL statements are executed, click the **Save** icon and then **Go to DataStudio for Scheduling** to schedule the node. You can enter the following sample SQL statements:

```
create table if not exists bank_data_holo_1_${bizdate} partition of bank_data_holo
 for values in ('${bizdate}');

insert into bank_data_holo_1_${bizdate}
select
   age as age,
   job as job,
   marital as marital,
   education as education,
   card as card,
    housing as housing,
   loan as loan,
   contact as contact,
   month as month,
   day_of_week as day_of_week,
    duration as duration,
   campaign as campaign,
    pdays as pdays,
   previous as previous,
   poutcome as poutcome,
    emp_var_rate as emp_var_rate,
   cons_price_idx as cons_price_idx,
   cons_conf_idx as cons_conf_idx,
   euribor3m as euribor3m,
   nr_employed as nr_employed,
   y as y,
   '${bizdate}' as ds
from bank_data_foreign_holo;
```

5. **Schedule the partitioned table.**

   Go to the DataStudio page and create a Hologres development node. In the SQL editor of the node, enter SQL statements to synchronize the partitioned table information to the node and click **Update Code**. Before you create the node, make sure that a workflow is created.

6. **Set parameters for scheduling the data analytics node.**

   On the editing tab of the Hologres development node, click the **Properties** tab in the right-side navigation pane to set parameters for scheduling the node.

   i. **Set parameters in the General section.**

      In the **Arguments** field, specify a value for the ${bizdate} variable.

ii. **Set parameters in the Schedule section.**

Select Normal for **Execution Mode** and set other parameters as required.

iii. **Set parameters in the Dependencies section.**

Select **Yes** for **Auto Parse** and click **Use Root Node**. After DataStudio automatically parses and displays the root node as a parent node, change the value of Auto Parse to **No**. You can also select a table that is scheduled as a parent node.

7. **Save and deploy the node for scheduling.**

After you set the scheduling parameters for the node, click the **Save** icon and then the **Submit** icon. After that, click **Deploy** in the upper-right corner.

8. **Deploy the node in Operation Center.**

On the **Create Package** page, find the target node and click Publish in the Actions column. After the node is deployed, click **Operation Center** in the top navigation bar to generate retroactive data for the node.

In **Operation Center**, right-click the published node and choose **Run** > **Current Node Retroactively**. Configure the node based on your business requirements.

9. **Check the content of the table in HoloStudio.**

After the retroactive data generation node is run, go back to HoloStudio. On the left-side navigation submenu, click **PG management**. On the PG management tab, click a database and choose Mode > public > **Table**. Double-click the partitioned table that is scheduled and click **Data Preview** to check whether the data is imported to the table.

2.6.5.3. Create multiple foreign tables at a time

Seamlessly integrated with MaxCompute at the underlying layer, Hologres allows you to create foreign tables to query MaxCompute data in an accelerated manner. You can create multiple foreign tables at a time by using the IMPORT FOREIGN TABLE statement. To free you from SQL operations, HoloStudio provides the following submodule for you to create foreign tables in a visualized manner: One-click MaxCompute table structure synchronization.

1. **Create a schema sync node.**

On the HoloStudio page, click **Data Analytics** on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **One-click MaxCompute table structure synchronization**. In the Create Node dialog box, set relevant parameters and click Commit. The schema sync node is created.

2. **Set parameters for synchronizing the table schema.**

After the schema sync node is created, you must set parameters for synchronizing the table schema based on your needs.

Parameter	Description	Remarks
Target Library	The name of the Hologres database where the foreign tables are to be created.	N/A

Parameter	Description	Remarks
Target Schema	The name of the schema in the specified Hologres database.	The default value is public. If you have created a schema, you can select the created schema.
Remote Service type	The type of the external service. You can create only foreign tables sourced from MaxCompute.	The default value is odps.
Remote server	The external server. The default value is odps_server.	After you purchase a Hologres instance, the system automatically creates a server named odps_server. You can directly use it.
Remote library	The name of the MaxCompute project where the tables mapping the foreign tables to be created reside.	N/A
Table name rules	The regular expression for specifying the tables whose schema is to be synchronized. By default, the schema of all tables in the specified MaxCompute project will be synchronized.	◦ If a foreign table to be created is named the same as an existing foreign table in Hologres, the foreign table is not created. ◦ If a MaxCompute table whose schema is to be synchronized contains data types that Hologres does not support, an error is thrown. In this case, exclude this MaxCompute table in the regular expression. ◦ For more information, see IMPORT FOREIGN SCHEMA.
Regular preview	The execution result of the regular expression.	N/A

3. **Run the schema sync node.**

   Click the **Save** icon and then click the **Run** icon to run the schema sync node. After the schema sync node is run, click **PG management** on the left-side navigation submenu. The created foreign tables appear. You can query the table data.

2.6.5.4. Import MaxCompute data

To improve the efficiency of querying MaxCompute data, Hologres allows you to import MaxCompute data to Hologres for queries. HoloStudio provides the following submodule for you to directly import MaxCompute data in a visualized manner: One-click MaxCompute data synchronization.

1. **Create a data sync node.**

   On the HoloStudio page, click Data Analytics on the left-side navigation submenu. On the Data Analytics tab, move the pointer over the Create icon and select **One-click MaxCompute data synchronization**. In the Create Node dialog box, enter the node information and click Commit. The data sync node is created.

2. **Set parameters for synchronizing data.**

   After the data sync node is created, you must set parameters for synchronizing data.

Section	Parameter	Description	Remarks
MaxCompute Source table selection	External table source	The source of the foreign table. Valid values: External table already exists and New external table.	◦ If you select External table already exists, the existing foreign table mapping the MaxCompute table will be used. ◦ If you select New external table, you must create a foreign table mapping the MaxCompute table.
	External table table name	The name of the existing foreign table.	The foreign table must map the MaxCompute table whose data will be synchronized.
Target table settings	Target Library	The name of the Hologres database to which the MaxCompute data will be synchronized.	N/A
	Target schema	The name of the schema in the specified Hologres database.	The default value is public. If you have created a schema, you can select the created schema.
	Destination Table Name	The name of the target table to which the MaxCompute data will be synchronized.	The table name can be customized.

Section	Parameter	Description	Remarks
	Target table description	The description of the target table.	N/A
Synchronization settings	Synchronization field	The fields to be synchronized from the specified MaxCompute table.	You can select specific or all fields in the MaxCompute table.
	Partition configuration	The partition fields to be synchronized.	Hologres supports a maximum of one level of partitions.
	Index configuration	The index to be built for the target table.	N/A
SQL Script	SQL Script	The SQL statements that are executed when the data sync node is run.	N/A

3. **Run the data sync node.**

   Click the **Save** icon and then click the **Run** icon to run the data sync node. After the node is run, you can query the imported data in SQL Console or PG management.

2.6.5.5. Upload local files

This topic describes how to upload local files in HoloStudio in a visualized manner.

Hologres allows you to use the COPY statement to import data from the standard input of a client to a specified table. For more information, see COPY. HoloStudio allows you to import data in a local file to a specified table by uploading the local file in a visualized manner. To upload a local file in HoloStudio, perform the following steps:

1. **Create a table.**

   In SQL Console or PG management, create a table to which data in the local file will be imported. In this example, use the following SQL statements to create a table:

```
BEGIN;
CREATE TABLE if not EXISTS holo_bank (
 age int8,
 job text,
 marital text,
 education text,
 card text,
 housing text,
 loan text,
 contact text,
 month text,
 day_of_week text,
 duration text,
 campaign int8,
 pdays float8,
 previous float8,
 poutcome text,
 emp_var_rate float8,
 cons_price_idx float8,
 cons_conf_idx float8,
 euribor3m float8,
 nr_employed float8,
 y int8
);
COMMIT;
```

2. **Create a node for uploading the local file.**

   Go to the HoloStudio page. On the left-side navigation submenu, click Data Analytics. On the Data Analytics tab, move the pointer over the Create icon and select **Upload files locally with one click.**

3. **Enter the node information.**

   In the **One-click local file Upload** dialog box, set the parameters based on your business needs and click Next Step.

Parameter	Description	Remarks
Target Library	The name of the Hologres database where the target table resides.	N/A

Parameter	Description	Remarks
Target Schema	The name of the schema where the target table resides.	The default value is public. If you have created a schema, you can select the created schema.
Select the data table to import	The name of the target table to which data in the local file will be imported.	N/A

4. **Select the local file to upload and set other required parameters.**

   After you click Next Step, select the local file to upload, set other required parameters, and then click Commit.

Parameter	Description	Remarks
Select File	The local file to upload.	You can select a local file only in the .txt, .csv, or .log format.
Select separator	The delimiter of fields in the file. Select Comma (,) or Space ( ).	N/A
Original character set	The character set of the file.	○ GBK ○ UTF-8 ○ CP936 ○ ISO-8859
First behavior title	Specifies whether to use the first line as the header line.	N/A

5. **View the imported data.**

   After you click **Commit**, data in the selected local file is imported to the specified table. You can go to SQL Console or PG management to view the imported data.

2.6.6. Hologres console

2.6.6.1. Overview

Hologres is a real-time interactive analytics service that is fully compatible with PostgreSQL and seamlessly integrated with the big data ecosystem. Hologres delivers high-concurrency and low-latency performance in analyzing terabyte-scale data. Hologres allows you to use mainstream Business Intelligence (BI) tools to get an analytical insight into data from multiple dimensions and explore business data in an efficient and cost-effective manner.

For convenience of business, Apsara Stack provides the Hologres console independent from the DataWorks console for different users to managing Hologres instances, users, and databases.

Log on to the Apsara Stack console. In the top navigation bar, choose **Products** > **Interactive Analytics** to go to the Hologres console. The following figure shows the Overview page in the Hologres console.

2.6.6.2. View the instance list

The Instances page lists all Hologres instances purchased by your Apsara Stack tenant account. On this page, you can view the instance status, change instance configurations, and create instances. You can also click an instance name to go to the instance details page where you can manage objects in the instance, including databases and users.

Instances

Log on to the Apsara Stack console. In the top navigation bar, choose **Products** > **Interactive Analytics** to go to the Hologres console. In the Hologres console, click **Instance List** in the left-side navigation pane. The following figure shows the Instance List page.

1. **New engine instance button**

   On the Instance List page, click **New engine instance**. In the New engine instance dialog box, enter an instance name and select the instance specifications to create a Hologres instance.

2. **Search box**

   If you have purchased multiple Hologres instances, you can enter a keyword of an instance name in the search box to find the target instance.

3. **Running status column**

   The Running status column displays the running status of each Hologres instance. An instance can be in one of the following states:

   - Normal operation: The instance is running as expected.
   - Creating: The payment is successful, and Hologres is creating the instance. You must wait for 3 to 5 minutes.
   - Shutdown: The instance has been suspended and you cannot connect to it.

Operation column

The Operation column provides the following buttons for you to manage a Hologres instance:

1. **Management**

   Find the target instance and click **Management** in the Operation column. On the page that appears, you can view and manage objects in the instance, including databases and users.

2. **Change configuration**

   If your instance cannot meet your business needs or your instance has a large amount of surplus resources, you can click **Change configuration** in the Operation column. In the **Change configuration** dialog box, upgrade or downgrade your instance configurations based on your business needs.

3. **Shutdown**

   Find the target instance and click **Shutdown** in the **Operation** column to suspend the instance. You cannot connect to the suspended instance.

2.6.6.3. Manage instances

This topic describes how to view and change instance configurations, select a network type, and select a connection method on the Basic information page in the Hologres console.

View and change instance configurations

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. The Basic information page displays basic information about a Hologres instance, including the instance name, instance ID, region, instance version, billing method, instance specification, and creation time.

If you need to change the instance specifications, click **Change configuration.** In the Change configuration dialog box, upgrade or downgrade the instance specifications based on your business needs.

Select a network type

The following table lists the supported network type.

Network type	Domain name	Scenario
Internal network	`<instancename>-cn-<region>-internal.hologres.aliyuncs.com:80`	Select this network type when you want to connect to the Hologres instance by using the classic network, without charges on the Internet traffic.

Select a connection method

Hologres is compatible with PostgreSQL. You can connect to a Hologres instance from the PostgreSQL client or over JDBC interfaces by using ETL or BI tools.

The Connection Methods section offers methods for you to use common development tools to connect to a Hologres instance. You can select a development tool and connection method based on your business needs and preference.

1. **Connect from the PostgreSQL client**

   To connect to a Hologres instance from the PostgreSQL client, use the following connection string:

   ```
   PGUSER=<AccessId> PGPASSWORD=<AccessKey> psql -p <Port> -h <Endpoint> -d <Database>
   ```

2. **Connect over JDBC**

   To connect to a Hologres instance over JDBC, use the following connection string:

   ```
   postgres://<AccessId>:<AccessKey>@<Endpoint>:<Port>/<database>? preferQueryMode=simple&tcpKeepAlive=true
   ```

2.6.6.4. Manage users

This topic describes how to manage users on the User Management page in the Hologres console.

Overview

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. On the page that appears, click **User Management**. On the User Management page, you can manage users on a Hologres instance without executing cumbersome SQL statements. For example, you can add and delete users and grant permissions to users on this page.

After you create a Hologres instance with your Apsara Stack tenant account, this account becomes a superuser of the instance. A superuser has all permissions on the Hologres instance. By default, the User Management page displays only the information of the Apsara Stack tenant account that creates the Hologres instance. The information of a Resource Access Management (RAM) user appears on this page only after you use the Apsara Stack tenant account to add it to the instance.

Column	Description	Remarks
Members	Displays the usernames of the Apsara Stack tenant account and RAM users on the Hologres instance.	Generally, a username appears in the xxx format.
Cloud account	Displays the account IDs of users on the Hologres instance.	N/A
Type	Displays the roles assigned to users on the Hologres instance.	The user can be a superuser or normal user.

Add a user

On the User Management page, you can create RAM users on a Hologres instance without executing the SQL CREATE statement.

Click **Add new user**. In the Add new user dialog box, select existing RAM users under your Apsara Stack tenant account to add them to the Hologres instance. If no RAM user exists under your Apsara Stack tenant account, create a RAM user first.

When you add a RAM user, you can assign the superuser or normal user role to the user.

- Superuser: A superuser has all permissions on the Hologres instance without the need for additional authorization.
- Normal user: A normal user cannot view or manage any objects on the Hologres instance, including databases, schemas, and tables. A normal user must be authorized before it can view and manage objects in the instance. We recommended that you go to the DB management page to grant permissions to RAM users as required. Alternatively, you can use SQL statements to grant permissions to RAM users.

Delete a user

Find the target user on the User Management page and click **Delete** in the Operation column to delete the user from the Hologres instance. A deleted user has no access to the Hologres instance.

2.6.6.5. Manage databases

This topic describes how to manage databases on the DB management page in the Hologres console.

Overview

In the Hologres console, click **Instance List** in the left-side navigation pane. In the instance list, find the target instance and click **Management** in the **Operation** column. On the page that appears, click **DB management**. On the DB management page, you can manage all databases on the current Hologres instance. You can create databases, select a permission management mode for the databases, and view database information.

> **Note** A default database named postgres is automatically created after you create a Hologres instance. This database is provided for management purposes only and does not appear on the DB management page. This database is allocated with limited resources. Create databases on this page based on your business needs.

Create a database

Hologres allows you to create a database with one click on the graphical user interface (GUI), eliminating the need for SQL operations.

Click **New Database**. In the New Database dialog box, enter a name for the database and set the Simple permissions model parameter to Open or Close. To simplify authorization, we recommend that you set the Simple permissions model parameter to Open.

Hologres provides two permission models for you to authorize users in a convenient way.

- Standard PostgreSQL authorization: Compatible with PostgreSQL, Hologres provides a permission model that is exactly the same as the standard PostgreSQL authorization model. You can authorize RAM users by using the standard PostgreSQL GRANT statement.
- SPM: Backed by the understanding of customers' business and its practical experience, Alibaba Cloud introduced a simple permission model (SPM) to Hologres to simplify the management of user permissions. The SPM is a coarse-grained model that authorizes users by user group.

After a database is created, you can use a development tool to connect to the database to analyze data.

Authorize a user

After the SPM is enabled for a new database, you can authorize RAM users with one click in the Hologres console. Perform the following steps:

1. **Open the Permission management right-side pane.**

   Find the target RAM user and click **User authorization** in the Operation column. You can grant permissions to a RAM user by adding the user to the desired user group.

2. **Add a RAM user to a user group.**

   In the Permission management right-side pane, click **Add authorization**. In the Add authorization dialog box, select the account to which you want to grant permissions, select the desired user group below Permissions policy, and then click **OK**.

Revoke permissions

If the SPM is enabled for your database, you can revoke the permissions of a RAM user with one click in the Hologres console.

On the instance details page, click **DB management**. On the DB management page, find the target database and click **User authorization** in the Operation column. In the Permission management right-side pane, find the target RAM user and click **Delete authorization** in the Operation column.

Delete a database

On the DB management page, find the database no longer required and click Delete in the Operation column to delete the database. After a database is deleted, data in the database is also deleted and cannot be recovered.

2.7. Realtime Analysis

2.7.1. Overview

Integrated with DataWorks, DataAnalysis supports creating MaxCompute tables in tabular mode, collaboratively editing workbooks and performing statistical analysis, and generating and sharing visual reports. These features enable data developers and business staff to quickly analyze data.

Go to DataAnalysis

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > DataAnalysis**.
3. On the **Home** page of DataAnalysis, click **Experience now** to go to the Web Excel page.

Features

- **Workbook**

  You can create and edit workbooks. Workbooks support basic operations such as addition, subtraction, multiplication, and division, and multiple data processing methods, including functions, classification, and aggregation. In addition, you can edit workbooks collaboratively with other users online and create pivot tables for further analysis.

- **Visual report**

  You can create and design visual reports by dragging, dropping, and configuring controls without running SQL statements.

- **Dimension table**

  You can create MaxCompute tables in tabular mode by one click without running SQL statements and edit MaxCompute tables collaboratively with other users online.

  - You can create a MaxCompute table in tabular mode.

- You can import data into a MaxCompute table by one click.

2.7.2. Workbook

2.7.2.1. Create a workbook

This topic describes how to create a workbook. After a workbook is created, you can rename, clone, delete, or change the owner of the workbook.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click  in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**.
4. On the **Web Excel** page, click  in the **New Spreadsheet** section.

   > **Note** If you have created workbooks, you can search for a workbook by entering its name in the search box in the **All Spreadsheets** section. Then, click the workbook name in the **File Name** column to go to the workbook editing page.

5. In the **New spreadsheet** dialog box, enter a name in the **File Name** field.
6. Click **OK**.

Result

After the workbook is created, it appears in the **All Spreadsheets** section. In this section, you can view all created workbooks. In addition, you can rename, clone, delete, or change the owner of a workbook.

- Find the target workbook and click **Rename** in the Operation column. In the **Rename** dialog box, enter the new name in the **File Name** field and click **OK**.
- Find the target workbook and click **Change Owner** in the Operation column. In the **Change Owner** dialog box, select an owner from the New Owner drop-down list and click **OK**.
- Find the target workbook and click **Clone** in the Operation column. The cloned workbook appears in the workbook list. The name of the cloned workbook contains the _copy suffix.
- Find the target workbook and click **Delete** in the Operation column. In the **Delete** message, click **OK**.

2.7.2.2. Edit a workbook

This topic describes how to edit a workbook. For example, you can import data to, export data from, and share a workbook, create a pivot table in a workbook, and use the data profiling feature.

Go to the workbook editing page

1. Log on to the DataWorks console.

2. On the DataStudio page, click in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**. The **Web Excel** page appears.
4. In the **All Spreadsheets** section of the **Web Excel** page, click the name of the target workbook in the **File Name** column.After you create a workbook, the workbook editing page appears.

Apply a template to a workbook or save a workbook as a template

You can apply an existing template to the current workbook by performing the following steps:

1. In the upper-right corner of the workbook editing page, choose **Template** > **Import Template**.
2. In the **Import Template** dialog box, select a file to be used as a template for the current workbook.

   > **Note** The data of the selected template will overwrite that of the current workbook.

3. Click **OK**.

You can save the current workbook as a template by performing the following steps:

1. In the upper-right corner of the workbook editing page, choose **Template** > **Save as Template**.
2. In the **Template settings** dialog box, set the **Type**, **Name**, and **Description** parameters.

   > **Notice** The template name can be up to 256 characters in length and the template description can be up to 1,024 characters in length.

3. Click **OK**.

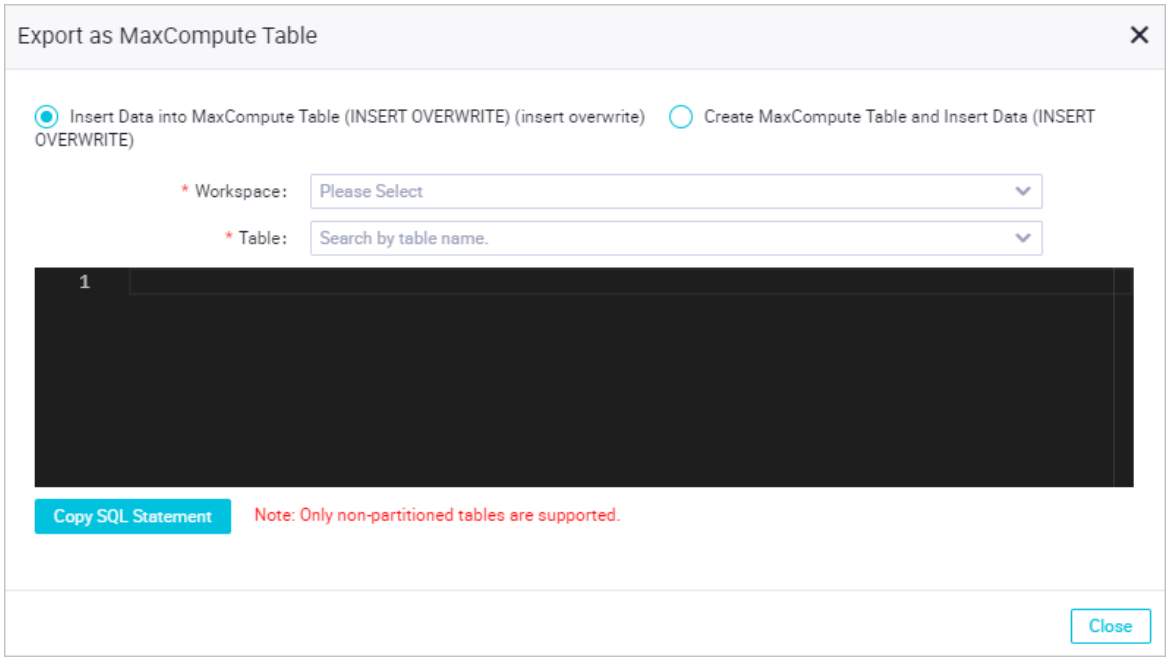
Import data to a workbook

1. In the upper-right corner of the workbook editing page, click **Import**.
2. In the Open dialog box, find and select a local file to be imported and click **Open**. The data in the local file is imported to the current workbook.

   > **Note** You can only import data from Excel files.

Export data from a workbook to a MaxCompute table

1. In the upper-right corner of the workbook editing page, choose **Export** > **Generate MaxCompute Build Table Statement**.
2. In the **Export as MaxCompute Table** dialog box, set relevant parameters.



Insert mode	Parameter	Description
Insert Data into MaxCompute Table (INSERT OVERWRITE) (insert overwrite)	**Workspace**	The workspace to which the MaxCompute table belongs.
	Table	The MaxCompute table to which you want to insert data.
Create MaxCompute Table and Insert Data (INSERT OVERWRITE)	**Workspace**	The workspace to which the MaxCompute table belongs.
	Table Name	The name of the MaxCompute table. Make sure that the table name has not been used. You can click **Check Duplicate Names** to check whether the table name exists.

3. After the parameters are set, click **Copy SQL Statement**.

> **Notice** Only non-partitioned tables are supported.

Create a pivot table

1. On the workbook editing page, select the data for which you want to create a pivot table and click **Pivot** in the upper-right corner.
2. In the **Create Pivot Table** dialog box, set relevant parameters.

   Specify the range of the data to be analyzed. You can set the **Choose Data** parameter to **Select Range** or **Use External Data Store** as needed.

   - If you select **Select Range**, select the cells in the workbook for which you want to create a

pivot table. The value of the **Range** field changes based on the selected cells.

> **Note** This parameter is available only when you select **Select Range**.

- If you select **Use External Data Store**, set the **Type** parameter first. You can set the Type parameter to **Mysql** or **Data Services**.
  - If you select **Mysql**, set the parameters described in the following table.

Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Select **Mysql**.
Workspace	The workspace where the MySQL data store resides.
Data Store	The name of the connection to the data store. To create a connection to a data store, perform the following steps: Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** in the left-side navigation pane. On the Data Source page, create a connection to a data store.
Table	The table for which you want to create a pivot table.

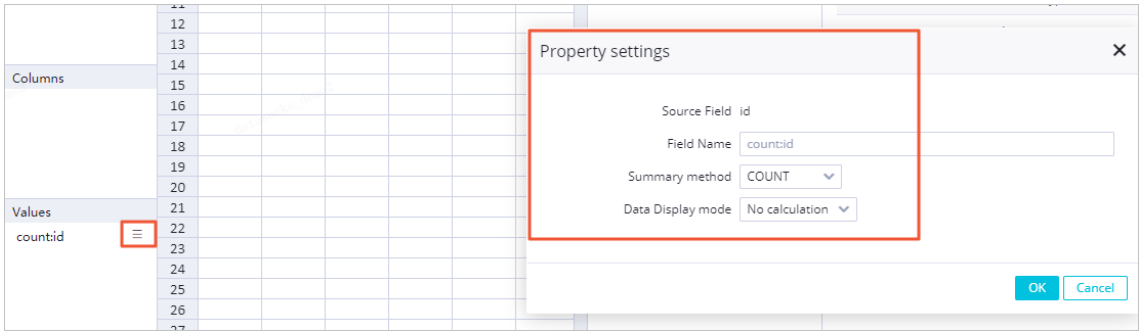
  - If you select **Data Services**, set the parameters described in the following table.

Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Select **Data Services**.
Workspace	The workspace where the API of DataService Studio resides.
API Group	The API group to which the API belongs.
API	The API to be used as the data source.

3. Click **OK**. The pivot table editing page appears.This topic takes the pivot table for a selected range of data as an example.
   - **Data Source**: the range that you specified in the previous step.
   - **Pivot Table Fields**: the names of the fields that you selected in the previous step.
   - **Rows**: Drag fields from the Pivot Table Fields section to the **Rows** section. Each value of

the field added to the **Rows** section occupies a row in the pivot table.

- **Columns**: Drag fields from the Pivot Table Fields section to the **Columns** section. Each value of the field added to the **Columns** section occupies a column in the pivot table.
- **Values**: Click the property setting icon for a field in the **Values** section. In the Property settings dialog box, set the **Summary method** and **Data Display mode** parameters. By default, the **Field Name** parameter cannot be modified.



Parameter	Description
Source Field	The name of the selected source field.
Field Name	The name of the field that appears in the pivot table. The name is in the format of **Aggregation method:Source field name**.
Summary method	The aggregation method. Valid values: **SUM**, **COUNT**, **MAX**, **MIN**, and **AVG**.
Data Display mode	The mode for displaying the data. Valid values: **No calculation** and **Percentage of Total**.

- **Filters**: Drag fields from the Pivot Table Fields section to the Filters section. In the right-side pivot table display area, you can select the fields to filter data.

Download a workbook

In the upper-right corner of the workbook editing page, click **Download** to download the workbook to a local directory.

Share a workbook

In the upper-right corner of the workbook editing page, click **Share**. In the dialog box that appears, set the sharing mode.

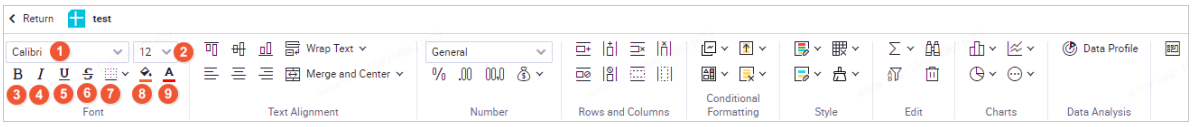
You can share the workbook in the following ways:

- **Link**: Click **Copy Link** and send the copied URL to other users as needed.
- **Users with Edit Access**: Click **Add** in the **Users with Edit Access** section. In the dialog box that appears, select the users to whom you want to grant the edit permission and click **OK**.
- **Visible to All**: To allow all users to view the workbook, turn on the **Visible to All** switch.
- **Users with Read Access**: To allow only specific users to view the workbook, turn off the **Visible to All** switch and click **Add** in the **Users with Read Access** section. In the dialog box that appears, select the users to whom you want to grant the read-only permission and click **OK**.

> **Note** If the system notifies you that the number of users to whom you want to grant the read-only permission reaches the upper limit, you can upgrade the DataWorks edition.

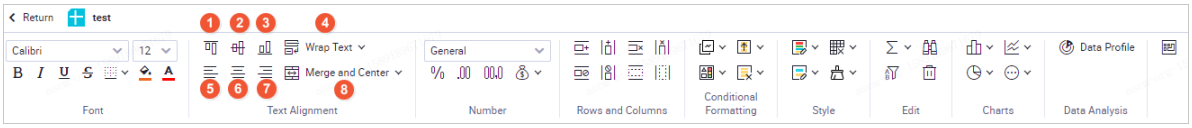
Menu bar

- **Font**



No.	Feature	Description
①	Font	Select a font type from the drop-down list as needed.
②	Font Size	Select a font size from the drop-down list as needed.
③	Bold	Set text in bold.
④	Italic	Set text in italic.
⑤	Underline	Underline text.
⑥	Strikethrough	Add a strikethrough to text.
⑦	Borders	Add borders to cells.
⑧	Background Color	Specify the background color of cells.
⑨	Text Color	Change the text color.

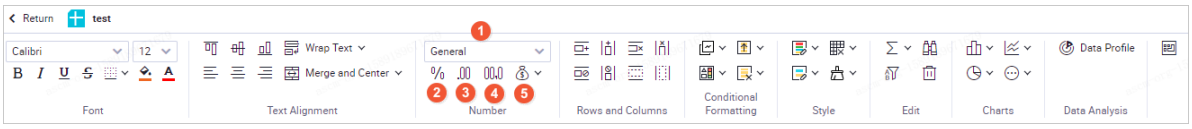
- **Text Alignment**



No.	Feature	Description
①	Top Align	Align text vertically to the top.
②	Middle Align	Align text vertically to the center.
③	Bottom Align	Align text vertically to the bottom.
④	Wrap Text	Display long text in multiple lines in a cell.

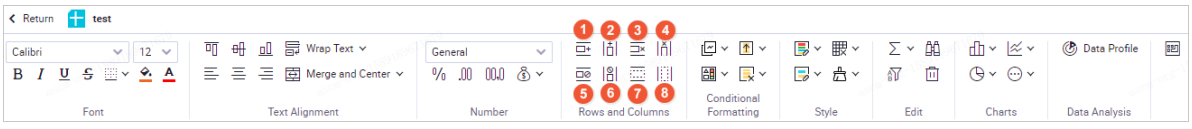
No.	Feature	Description
⑤	Align Left	Align text horizontally to the left.
⑥	Center	Align text horizontally to the center.
⑦	Align Right	Align text horizontally to the right.
⑧	Merge and Center	Merge multiple cells to one cell and center the content in the cell.

- **Number**



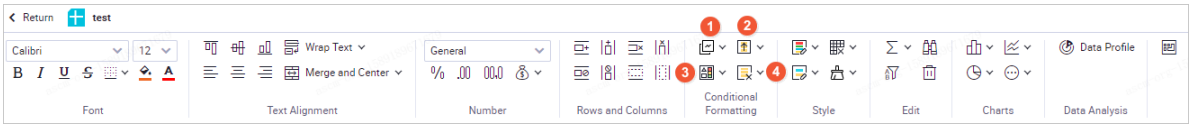
No.	Feature	Description
①	Data Type	Specify the type of data held in cells. You can select General, Number, Currency, Short Date, Long Date, Time, Percentage, Fraction, Scientific, and Text from the drop-down list.
②	Percentage	Apply the percentage format to numbers.
③	Two Decimal Places	Round numbers to two decimal places.
④	1000 Separator	Display numbers with thousands separators, for example, 1,005.
⑤	Currency	Add a currency sign to numbers. The following currency signs are supported: yuan sign (¥), dollar sign ($), pound sign (£), euro sign (€), and franc sign (Fr).

- **Rows and Columns**



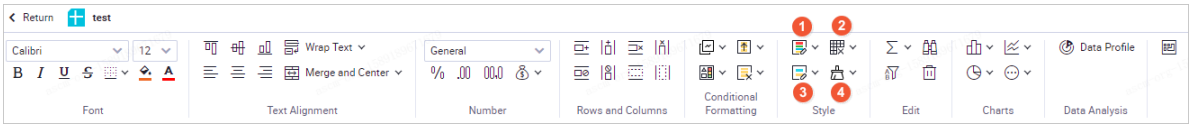
No.	Feature	Description
①	Insert Row	Insert a row to the workbook.
②	Insert Column	Insert a column to the workbook.
③	Delete Row	Delete rows from the workbook.
④	Delete Column	Delete columns from the workbook.
⑤	Lock Row	Lock the rows before the selected row in the workbook.
⑥	Lock Column	Lock the columns before the selected column in the workbook.
⑦	Hide Row	Hide rows in the workbook.
⑧	Hide Column	Hide columns in the workbook.

- **Conditional Formatting**



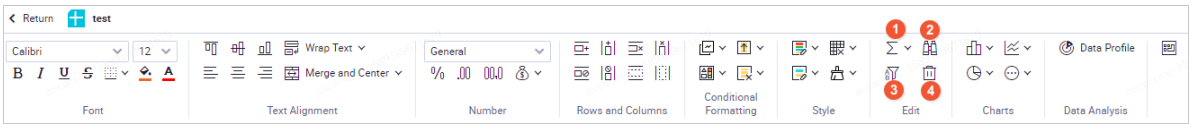
No.	Feature	Description
①	Highlight cell rules	Specify the rules for highlighting cells.
②	Data Bar/Color Scale	Format cells by using data bars and color scales.
③	Icon Set	Format cells by using icon sets. The icon sets include directional icons, shapes, indicators, and rating icons.
④	Clear Rule	Clear the formatting. You can select **Clear Rules from Selected Cells** or **Clear Rules from Entire Sheet** from the drop-down list.

- **Style**



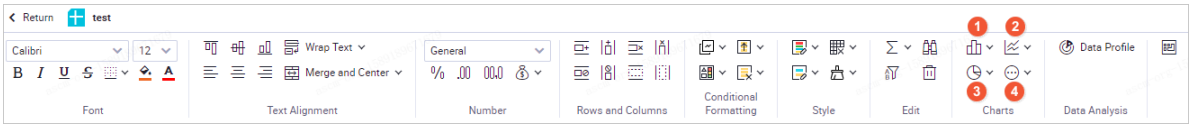
No.	Feature	Description
①	Apply table style	Apply a predefined table style to cells.
②	Delete	Remove the applied table style.
③	Cell Style	Apply a cell style to cells.
④	Clear	Clear the content or style in cells. You can select **Clear All**, **Clear Content**, or **Clear Style** from the drop-down list.

- **Edit**



No.	Feature	Description
①	AutoSum	Select an aggregation method. You can select **Sum**, **Average**, **Count Numbers**, **Max**, or **Min** from the drop-down list.
②	Search	Click **Search** or press **Ctrl+F** to open the search box.
③	Sort and Filter	Filter data and sort data in ascending or descending order.
④	Clear	Clear the content in cells.

- **Charts**

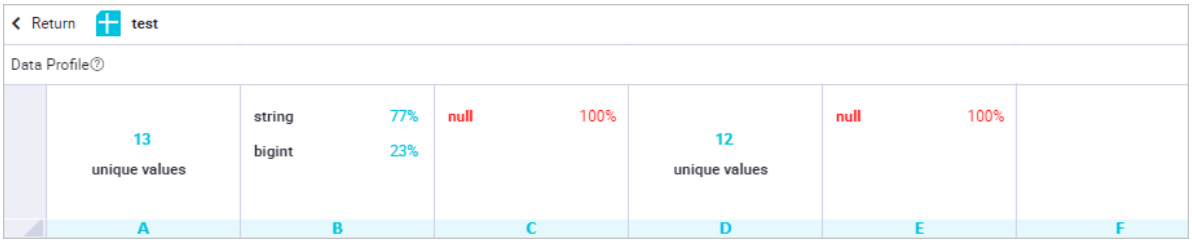


  - **Column Chart**: After you click the **Column Chart** icon, you can select **Column Chart**, **Stacked Column Chart**, or **100% Stacked Column Chart**.
  - **Line Chart**: After you click the **Line Chart** icon, you can select **Line Chart**, **Stacked Line Chart**, **100% Stacked Line Chart**, **Line Chart with Markers**, **Stacked Line Chart with Markers**, or **100% Stacked Line Chart with Markers**.
  - **Pie Chart**: After you click the **Pie Chart** icon, you can select **Pie Chart** or **Doughnut Chart**.
  - **More**: After you click the **More** icon, you can view more chart types, including area charts, bar charts, scatter charts, and stock charts.

- **Data Profile**

The data profiling feature allows you to analyze the quality, structure, distribution, and statistics of the data. It also allows you to preview, profile, process, analyze, and visualize data. The data profiling feature analyzes data by column and allows you to view the distribution of data types and values of each column.

Select the data to be analyzed and click **Data Profile**. The data profiling feature displays the data type and value distribution of each column above the editing area in the form of charts and rich text.



Simple mode:

- For a column whose values are of the STRING or DATE type: The simple mode displays the values ranking top 2 based on frequency and their respective percentages, and the percentage of other values in the form of rich text. If the number of value types exceeds 50% of the total number of values, the simple mode displays the number of unique values.
- For a column whose values are of the INTEGER or FLOAT type: The simple mode displays the value distribution in the form of a histogram.
- For a column whose values are of the BOOLEAN type: The simple mode displays the proportions of different values in the form of pie charts.
- For a column whose values are of two or more data types: The simple mode displays the proportions of different data types in the form of pie charts. In addition, the system reminds you that the current column has dirty data. After the dirty data is cleared, the simple mode displays the data in one of the preceding forms based on the data type.
- For a column whose values are null values: The simple mode displays the percentage of null values in red.

Click **Detailed Mode** in the upper-right corner. In the **Data Profile** dialog box, you can view the profiling result of each column.

Detailed mode:

- For a column whose values are of the STRING or DATE type: The detailed mode displays the number of fields, the numbers and percentages of unique values, valid values, and null values, and the numbers of occurrences of the values ranking top 5 based on frequency.
- For a column whose values are of the INTEGER or FLOAT type: The detailed mode displays the number of fields, the numbers and percentages of unique values, valid values, zeros, and null values, the numbers of occurrences of the values ranking top 5 based on frequency, the statistics, and a histogram.
- For a column whose values are of the BOOLEAN type: The detailed mode displays the number of fields, the numbers and percentages of unique values, zeros, and null values, the numbers of occurrences of the values ranking top 5 based on frequency, and a pie chart.

> **Note** The system considers the true and false strings and the 0 and 1 integers as values of the BOOLEAN type.

- **List of Shortcut Keys**

  Click  to view the shortcut keys for different features.

2.7.3. Report

2.7.3.1. Create a report

This topic describes how to create a report. After a report is created, you can rename or delete the report.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page, click  in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. On the DataAnalysis homepage, click **Experience Now**.
4. On the **Web Excel** page, click **Report** in the top navigation bar.
5. Click in the **New Report** section.Alternatively, click a template in the **New Report** section to create a report.

   > **Note** If you have created reports, you can search for a report by entering its name in the search box in the **All Reports** section. Then, click the report name in the **File Name** column to go to the report editing page.

6. In the **New Report** dialog box, set the **Report Name** and **Report Description** parameters.
7. Click **OK**.

Result

After the report is created, it appears in the **All Reports** section. In this section, you can view all created reports. In addition, you can rename or delete a report.

- To rename a report, perform the following steps: Find the target report and click **Rename** in the Operation column. In the **Rename** dialog box, enter the new name in the **File Name** field and click **OK**.
- To delete a report, perform the following steps: Find the target report and click **Delete** in the Operation column. In the **Delete** message, click **OK**.

2.7.3.2. Edit a report

This topic describes how to edit, preview, save, share, and release a report, and save a report as a template.

Procedure

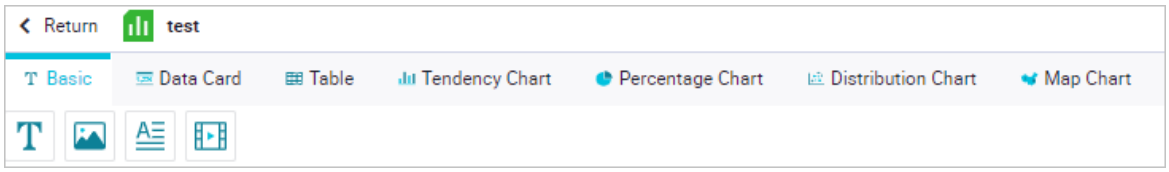
1. Go to the Report page.
   i. Log on to the DataWorks console.

ii. On the DataStudio page, click in the upper-left corner and choose **All Products** > **DataAnalysis**.

iii. On the DataAnalysis homepage, click **Experience Now**.

iv. On the **Web Excel** page, click **Report** in the top navigation bar.

2. Go to the report editing page.Use one of the following methods to go to the report editing page:
   - After you create a report, the report editing page appears.
   - In the **All Reports** section of the Report page, click the name of the target report in the **File Name** column.
3. Drag controls from the menu bar to the canvas. In this topic, the **Bar Chart** control on the **Tendency Chart** tab is dragged to the current report.You can drag a control from the menu bar to the canvas to use the control as a component in the report.

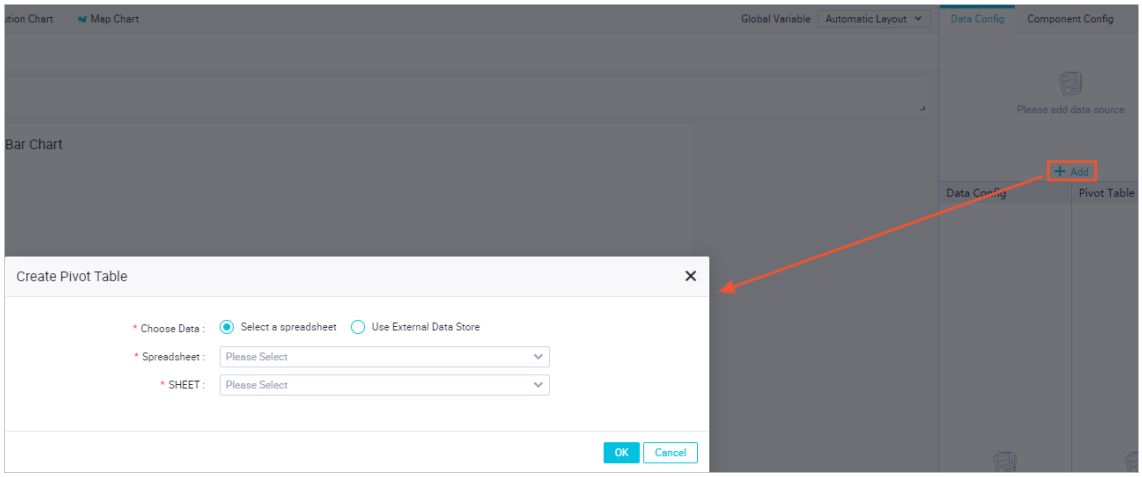


4. Click the **Bar Chart** component. Then, select a data store on the **Data Config** tab of the configuration section.If the required data store is added, click the name of the data store on the **Data Config** tab.

   If you need to add a data store, click **Add** on the **Data Config** tab. In the **Create Pivot Table** dialog box, set relevant parameters and click **OK**.

   You can set the Choose Data parameter to Select a spreadsheet or Use External Data Store.

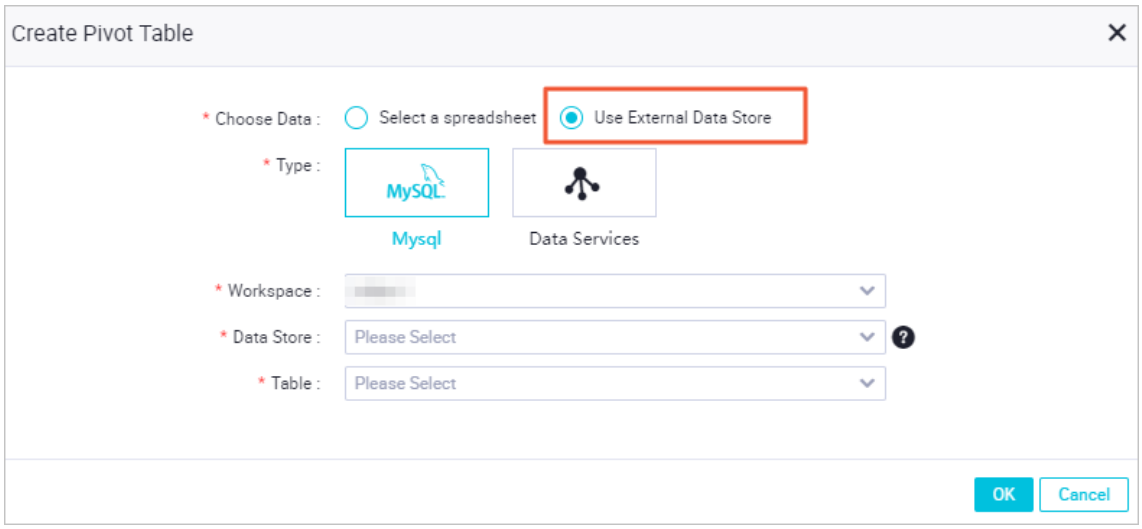
   - **Select a spreadsheet**: You can select an editable worksheet in a workbook of the current user as the data store.



Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Select a spreadsheet**.

Parameter	Description
Spreadsheet	The workbook from which the data is analyzed. Select a workbook from the **Spreadsheet** drop-down list.
SHEET	The worksheet of which the data is analyzed. Select a worksheet from the **SHEET** drop-down list.

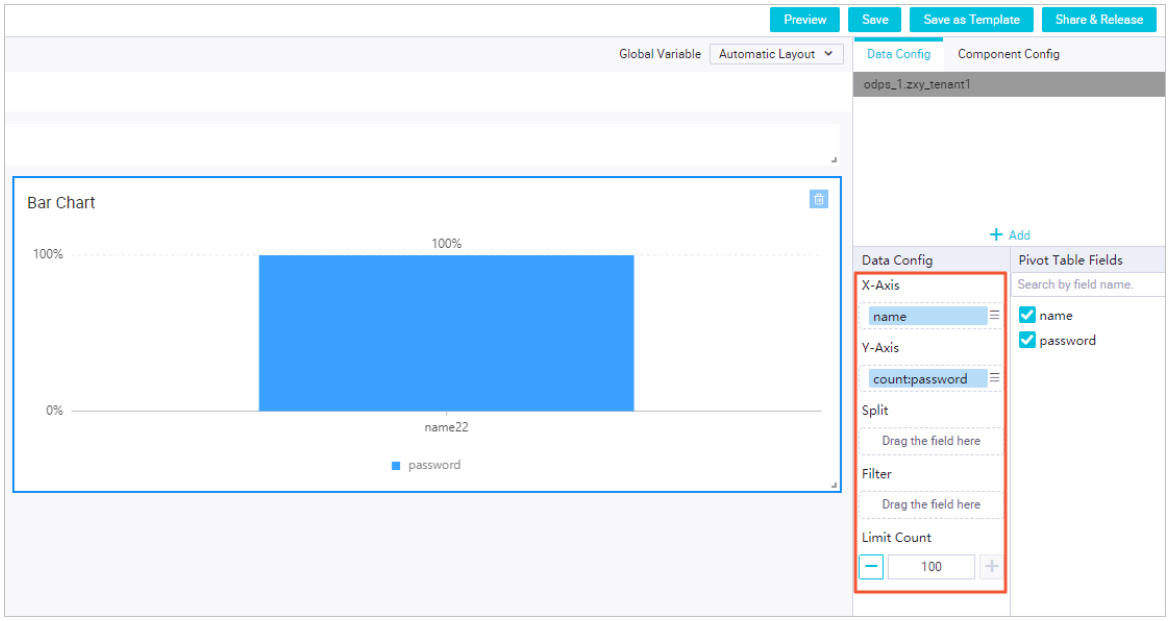
- **Use External Data Store**: You can select a MySQL data store or the API of DataService Studio.



Parameter	Description
Choose Data	The range of the data to be analyzed. Select **Use External Data Store**.
Type	The type of the data source. Valid values: **Mysql** and **Data Services**.
Workspace	The workspace where the MySQL data store resides.
Data Store	The name of the connection to the data store. To create a connection to a data store, perform the following steps: Click the **Workspace Manage** icon in the upper-right corner. On the page that appears, click **Data Source** in the left-side navigation pane. On the Data Source page, create a connection to a data store. **Note** This parameter is available only when you set the Type parameter to **Mysql**.

Parameter	Description
Table	The table of which the data is analyzed. Note This parameter is available only when you set the Type parameter to Mysql.
API Group	The API group to which the API belongs. Note This parameter is available only when you set the Type parameter to Data Services.
API	The API to be used as the data source. Note This parameter is available only when you set the Type parameter to Data Services.

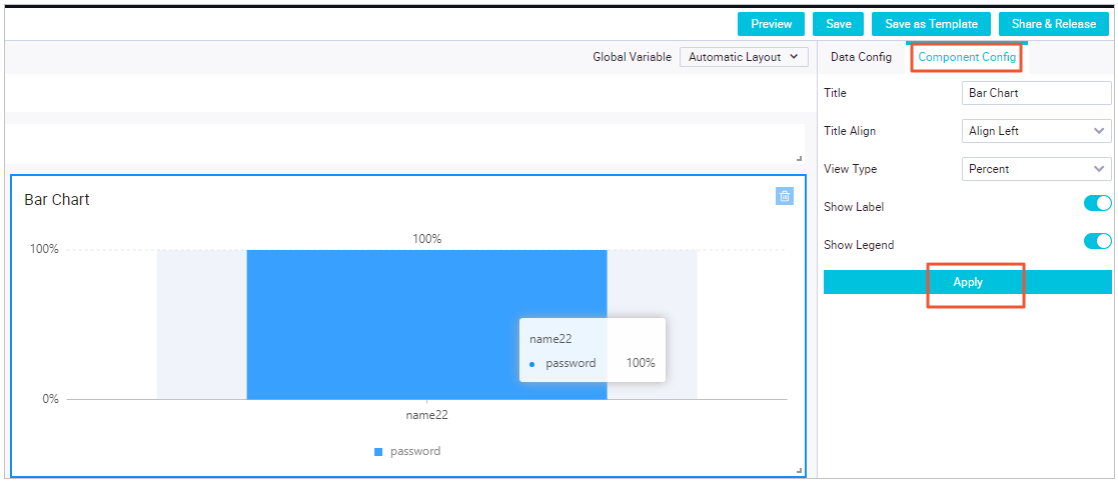
5. Select fields as statistical items.The fields that are required vary with the chart type. For example, you must specify the X-axis, Y-axis, split, and filter fields for a column chart by dragging fields from the **Pivot Table Fields** section to the **Data Config** section. You can also specify the number of vertical columns that can appear in the column chart.



Multiple charts can use the same data store. They can use the data store in different ways without affecting each other. A chart can use only one data store. When you click a chart and drag fields from the Pivot Table Fields section to the Data Config section, the chart is associated with the data store.

6. Configure the information about the column chart.
   i. On the canvas, click the **Bar Chart** component.
   ii. Click the **Component Config** tab in the right-side configuration section.

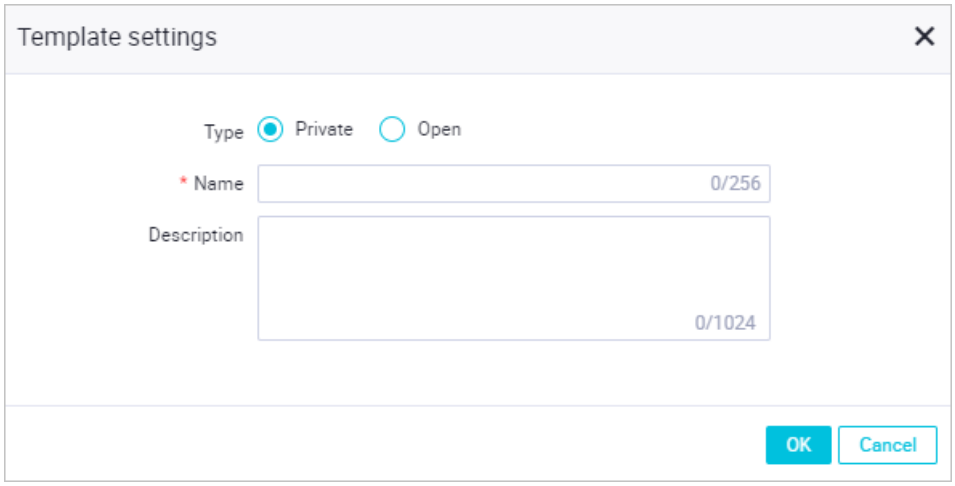
iii. On the **Component Config** tab, set relevant parameters.



Parameter	Description
Title	The title of the component.
Title Align	The alignment of the chart title. Valid values: **Align Left**, **Align Center**, and **Align Right**.
View Type	The display mode of vertical columns. Valid values: **Stack**, **Parallel**, and **Percent**.
Show Label	Specifies whether to display labels for the component.
Show Legend	Specifies whether to display legends for the component.

7. Click **Apply**.
8. Return to the **Report** page or preview, save, share, or release the report as needed.
   - **Return**: You can click **Return** in the upper-left corner to return to the **Report** page. On the Report page, you can view other reports and go to the report editing page of other reports.
   - **Preview**: You can click **Preview** in the upper-right corner to preview the report.
   - **Save**: You can click **Save** in the upper-right corner to save the report, so that you can open and edit the saved report next time.
   - **Save as Template**: You can save a report as a template, so that you can create a report based on the template. Perform the following steps to save a report as a template:
     a. On the report editing page, click **Save as Template** in the upper-right corner.
     b. On the **Preview** page, click **Next Step (Template setup)**.

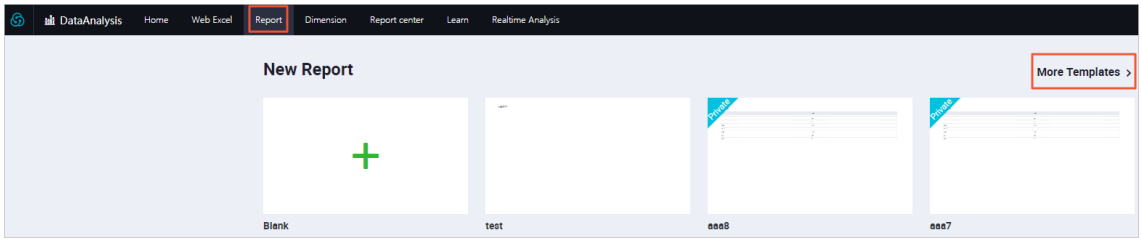
c. In the **Template settings** dialog box, set relevant parameters.



Parameter	Description
Type	Specifies whether to show or hide the template for other users. Valid values: **Private** and **Open**.
Name	The name of the template. The name can be up to 256 characters in length.
Description	The description of the template. The description can be up to 1,024 characters in length.

d. Click **OK**.

After you save the report as a template, you can click the template in the New Report section of the **Report** page to create a report.



- **Share & Release**: You can click **Share & Release** in the upper-right corner to share and release the report. You can share the report with specified users or all users. If you need to share the report to specific users, click **Add** to specify the users.

2.7.4. Go to the report center

On the Data analysis report center page, you can view the visualized report solution that DataAnalysis provides.

Procedure

1. Log on to the DataWorks console.

2. On the **DataStudio** page, click in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. In the top navigation bar of the DataAnalysis page, click **Report center** to go to the **Data analysis report center** page.

2.7.5. Go to the learning center

On the Data analytics Learning Center page, you can view relevant materials and learn about DataAnalysis.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click in the upper-left corner and choose **All Products** > **DataAnalysis**.
3. In the top navigation bar of the DataAnalysis page, click **Learn** to go to the **Data analytics Learning Center** page.

2.8. Administration

2.8.1. Overview

Generally, developers need to test workflows and nodes on the Operation Center page.

As a key tool for routine O&M, Operation Center enables you to manage and maintain the workflows and nodes that you have committed. The Operation Center service consists of four modules: Dashboard, Nodes, Node Instances, and Monitor.

- **Dashboard**: enables you to view and manage all global nodes of DataWorks. It displays various information, including **Instances**, **Instances Run Today**, **Node Runtime**, **Instances Run in the Last Month**, **Nodes with Errors in the Last Month**, and **Node Types** of the current workspace.
- **Nodes**: provides **Recurring** and **Manually Triggered**.
- **Node Instances**: provides **Recurring**, **Manually Triggered**, **Smoke Test** and **Retroactive**. You can manage them in a list view or DAG.
  - The list view displays the running status of nodes in a list. You can add multiple alerts at a time, change owners, and add nodes to baselines.
  - In the DAG, you can maintain and manage the running status of nodes and their dependencies on ancestor and descendant nodes. You can also perform operations, such as retroactive data generation and rerun, for a single node.
- **Monitor**: provides **Baseline Instances**, **Baselines**, **Events**, **Alert Triggers**, and **Alerts**.

2.8.2. Dashboard

The Dashboard page provides information about the node running status, the trend of the number of nodes that were run, the node running time, and the nodes with errors.

Go to the Dashboard page

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Operation Center**. By default, the **Dashboard** page appears.

The **Dashboard** page consists of the following sections: **Instances**, **Instances Run Today**, **Node Runtime**, **Nodes with Errors In the Last Month**, **Instances Run In the Last Month**, and **Node Types**.

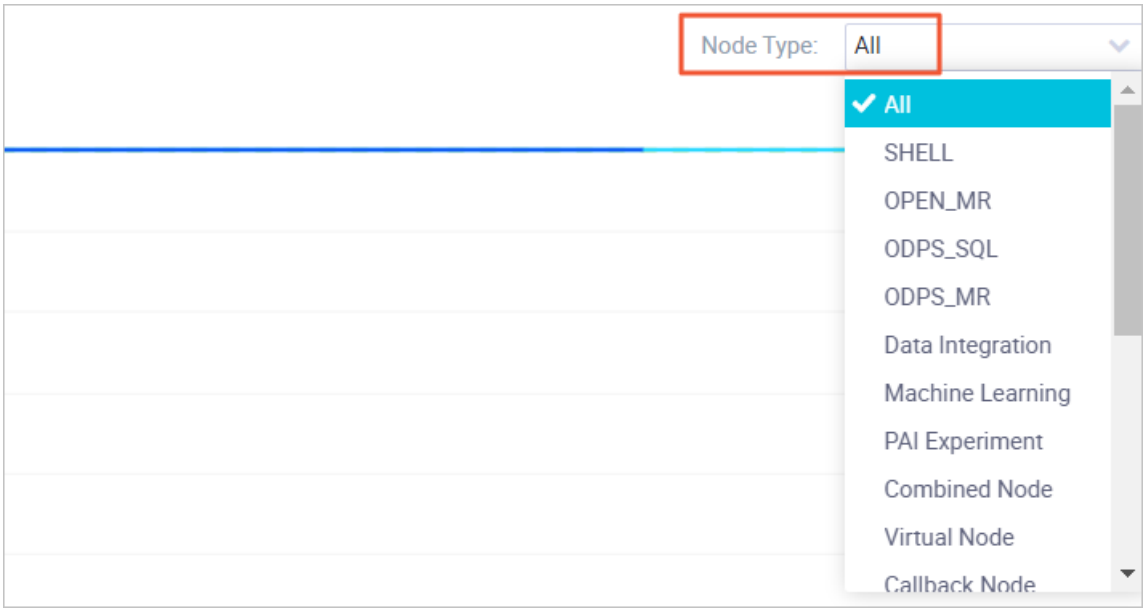
> **Note**
> - Pending (Schedule): the instances whose scheduled time has not arrived. These instances will be automatically run when their scheduled time arrives.
> - Pending (Resources): the instances whose scheduled time has arrived. These instances are waiting for scheduling resources.

View the summary of instance running

The **Instances** section displays the numbers of auto triggered node instances that were run today and yesterday, as well as the historical average. If the deviations among the three numbers are large, an exception occurred during a specific period of time. Further check and analysis are required.

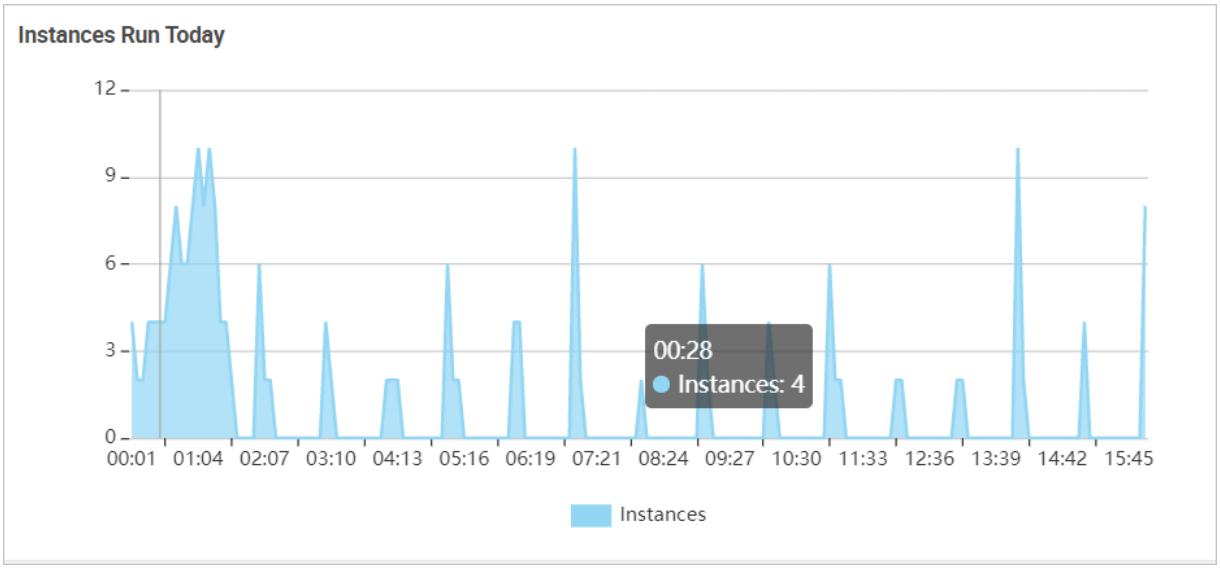
The statistical chart on the right of the Instances section shows three lines in different colors, representing the numbers of instances completed between 00:00 and 23:00 of today and yesterday, as well as the historical average. You can also view the number and proportion of instances in the pie chart on the left.

In the upper-right corner of the statistics chart, you can select a node type from the **Node Type** drop-down list to view statistics on the specified type of nodes.



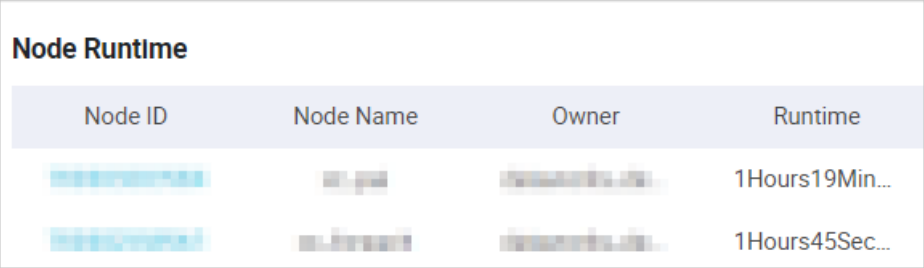
View the statistics on instances run today

The **Instances Run Today** section displays the numbers of node instances that were running at different time points of the current day. In the chart, you can find the time point when peak concurrency occurred and the maximum number of concurrent nodes. Based on the information, you can determine whether to avoid node scheduling at the peak hours and adjust the scheduling time of nodes.



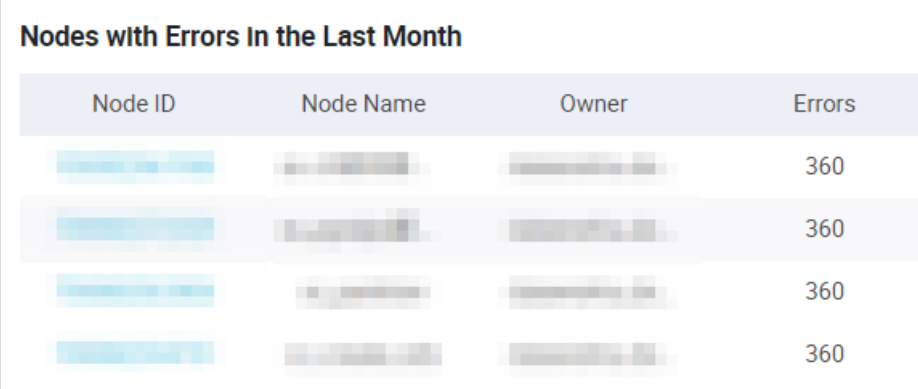
View the rankings of nodes based on their running time

The **Node Runtime** section displays nodes with a specified timestamp in the current workspace by their running time. By default, this section displays the top 10 nodes in descending order of their running time. You can view **Node ID**, **Node Name**, **Owner**, and **Runtime** of each node.



View the rankings of nodes with errors in the last month

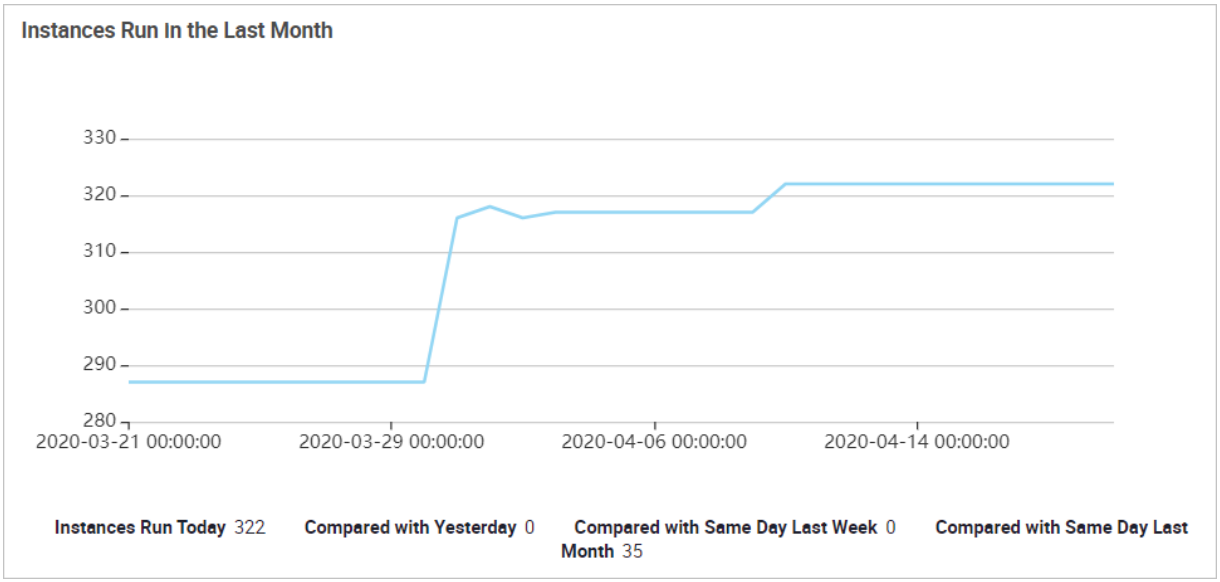
The **Nodes with Errors In the Last Month** section displays the top 10 nodes with the most errors in the last month. You can view **Node ID**, **Node Name**, **Owner**, and **Errors** of each node.



You can click a node ID to go to the node details page.

View the trend of the number of instances that were run

The **Instances Run In the Last Month** section displays the number of instances that were run today, comparison with yesterday, comparison with the same day last week, and comparison with the same day last month.



View the node distribution by node type

In the **Node Types** section, you can move the pointer over a sector of the pie chart to view the number and proportion of nodes of the specific type.

2.8.3. Auto triggered node O&M

2.8.3.1. Manage auto triggered nodes

Auto triggered nodes are automatically run as scheduled after they are committed to the scheduling system.

> **Note**
>
> - By default, the auto triggered node list displays nodes in all the workflows created by the current account.
> - After you commit a node, the scheduling system automatically generates and runs an instance of the node at 23:30 the next day. If you commit a node after 23:30, the scheduling system generates and runs an instance of the node on the third day.
> - Do not perform any operations on the **Workspace name_root** node, which is the root node of the workspace. All instances of auto triggered nodes depend on this node. If this node is frozen, instances of auto triggered nodes cannot be run.

Manage auto triggered nodes in the node list

The Cycle Task page displays auto triggered nodes that are committed to the scheduling system in a list.

Operation	Description
Filter	Find required nodes by setting parameters in the red box marked with 1 in the preceding figure. You can search for nodes by node name or node ID and set the **Node Type**, **Owner**, **My Nodes**, **Modified Today**, and **Frozen** parameters to filter nodes. **Note** When you search for nodes by node name, the search result is affected by other filter conditions you specified. Only the nodes that meet both the specified search condition and other filter conditions are returned in the search result.
DAG	Click **DAG** in the Actions column of a node to view the directed acyclic graph (DAG) of the node. You can view the node information, such as properties, operations logs, and code, in the DAG.
Test	Click **Test** in the Actions column of a node to test the node. For more information, see Manage test instances.
Patch Data	Click **Patch Data** in the Actions column of a node and select an item from the drop-down list to generate retroactive data for the node. For more information, see Manage retroactive instances.

Operation	Description
More operations	Click **More** in the Actions column of a node to perform more operations on the node. You can perform the following operations on the node: • Select **Freeze** to freeze the node. After the node is frozen, DataWorks can still generate instances of the node, but does not run the instances of the node and its descendant instances. • Select **Unfreeze** to unfreeze the node. After the node is unfrozen, DataWorks can normally run the instances of the node and its descendant instances. • Select **View Instances** to view the instances of the node. • Select **Configure Alert Trigger** to configure alert triggers for the node. • Select **Change Owner** to change the owner of the node. • Select **Add to Baseline** to add the node to a baseline. • Select **Change Resource Group** to change the resource group used to run the node if multiple resource groups exist in the workspace. • Select **Configure Data Quality Rules** to configure rules for monitoring the data quality of the node. • Select **View Lineage** to view the lineage of the node. • Select **View Ancestor and Descendant Nodes** to go to the **Node Information** page, where you can view node information on the **Ancestor Nodes** and **Descendant Nodes** tabs.
Batch operations	Select multiple nodes and click a button in the red box marked with 3 in the preceding figure to perform batch operations. The buttons include **Configure Alert Trigger**, **Change Owner**, **Change Resource Group**, **Add to Baseline**, **Freeze**, **Unfreeze**, and **Delete**.

Manage auto triggered nodes in a DAG

Click the name of a node or **DAG** in the Actions column to view the DAG of the node. In the DAG, you can right-click the node to perform related operations.

Operation	Description
Show Ancestor Nodes or **Show Descendant Nodes**	View ancestor or descendant nodes at one or more levels. If the workflow contains three or more nodes, the DAG displays only the current node and hides its ancestor and descendant nodes.
View Node Details	Go to the **Node Information** page to view the node information, including the input table, output table, ancestor nodes, and descendant nodes.
View Code	View the code of the node.
Edit Node	Go to the DataStudio page to modify the node.
View Instances	View the instances of the node.
View Lineage	View the lineage of the node.

Operation	Description
Test	Go to the **Smoke Test** dialog box. Set the **Smoke Test Instance Name** and **Data Timestamp** parameters and click **OK**. Then, the Test Instance page appears.
Run	Select **Current Node Retroactively**, **Current and Descendant Nodes Retroactively**, or **Mass Nodes Retroactively**.
Freeze	Pause the scheduling of the node.
Unfreeze	Resume the scheduling of the node after it is frozen.
Configure Data Quality Rules	Configure rules for monitoring the data quality of the node.

2.8.3.2. Manage auto triggered node instances

Auto triggered node instances are snapshots taken for auto triggered nodes that are run.

An instance is generated each time when an auto triggered node is run as scheduled. You can manage auto triggered node instances. For example, you can view the running status of instances, and stop, rerun, and unfreeze instances.

> **Note** DataWorks regularly generates instances for auto triggered nodes. Each generated instance runs the latest code. If you modify and recommit the node code after instances are generated, the instances that have not been run will run the latest code.

Manage auto triggered node instances in the instance list

You can manage auto triggered node instances in the instance list. For example, you can check operational logs, rerun instances, and stop running instances.

Operation	Description
Filter	Find required instances by setting parameters in the red box marked with 1 in the preceding figure. You can search for instances by node name or node ID and set the following parameters to filter instances: **Data Timestamp**, **Node Type**, **Run At**, **Solution**, **Workflow**, **Region**, **Engine type**, **Engine instance**, **Baseline**, **Owner**, **Recurrence**, **Status**, **My Nodes**, **My Nodes with Errors**, **My Incomplete Nodes**, and **Re-run node**. > **Note** By default, the data timestamp is set to the previous day of the current day.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.

Operation	Description
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. Note This operation applies only to instances in the Pending (Ancestor), Successful, or Failed state.
Rerun Descendant Nodes	Rerun the instance and its descendant instances. You must specify the instances to rerun. After they are run, their descendant instances will be run as scheduled. This operation applies to data recovery. Note You can select instances only in the Pending (Ancestor), Successful, or Failed state. The value No appears in the Meet Rerun Condition column of instances in other states, and you cannot select the instances.
Set Status to Successful	Set the status of the instance to Successful and run its descendant instances as scheduled. You can perform this operation if an instance fails. Note This operation applies only to failed instances but not workflows.
Freeze	Freeze the instance if it is in the Running state.
Unfreeze	Unfreeze the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.
Batch operations	Select multiple instances and click a button in the red box marked with 3 in the preceding figure to perform batch operations. The buttons include Stop, Rerun, Set Status to Successful, Freeze, and Unfreeze.

Manage auto triggered node instances in a DAG

Click the name of an instance or DAG in the Actions column to view the directed acyclic graph (DAG) of the instance. In the DAG, you can right-click the instance to perform related operations.

Operation	Description
Show Ancestor Nodes or Show Descendant Nodes	View ancestor or descendant instances at one or more levels. If a workflow contains three or more instances, the DAG displays only the current instance and hides its ancestor and descendant instances.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the DataStudio page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
More operations	View more instance information on the **General**, **Context**, **Runtime Log**, **Operation Log**, and **Code** tabs.
Stop	Stop the instance if it is in the Pending (Resource) or Running state. After you perform this operation, the instance enters the Failed state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. **Note** This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
Rerun Descendant Nodes	Rerun the instance and its descendant instances. You must specify the instances to rerun. After they are run, their descendant instances will be run as scheduled. This operation applies to data recovery. **Note** You can select instances only in the **Pending (Ancestor)**, **Successful**, or **Failed** state. The value No appears in the **Meet Rerun Condition** column of instances in other states, and you cannot select the instances.
Set Status to Successful	Set the status of the instance to Successful and run its descendant instances as scheduled. You can perform this operation if an instance fails. **Note** This operation applies only to failed instances but not workflows.
Resume	Continue to run the instance after it fails.
Emergency Operations	Perform emergency operations, including **Delete Dependencies**, **Change Priority**, and **Force Rerun**. Perform these operations in emergency only. These operations take effect only on the current instance for one time. Select **Delete Dependencies** to delete dependencies of the current instance. You can perform this operation so that you can start the current instance when the ancestor instances fail and the current instance does not depend on the data of the ancestor instances.

Operation	Description
Freeze	Freeze the instance if it is in the Running state.
Unfreeze	Unfreeze the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.

2.8.3.3. Manage retroactive instances

DataWorks runs retroactive instances to generate retroactive data for auto triggered nodes. You can manage retroactive instances. For example, you can view the running status of instances, and stop, rerun, or unfreeze instances.

Limits

- When DataWorks generates retroactive data in a period of time for a node, if one instance of the node fails on a day, the retroactive instance for that day is also set to Failed. DataWorks will not run instances of this node for the next day. To sum up, DataWorks runs instances of a node on a day only when all its instances of the previous day are successful.
- For a self-dependent auto triggered node, if the first instance for which retroactive data needs to be generated has a last-cycle instance on the previous day but the last-cycle instance is not run, the retroactive instance cannot be triggered. If the first instance for which retroactive data needs to be generated does not have a last-cycle instance on the previous day, the retroactive instance is directly triggered.
- DataWorks generates alerts only for auto triggered node instances.
- If an auto triggered node has an instance in the Running state, its retroactive or test instance can start to run only after this auto triggered node instance is run.
- If both an auto triggered node instance and a retroactive instance are running for a node at the same time, you must stop the retroactive instance to ensure proper running of the auto triggered node instance.

Go to the page for configuring retroactive instances

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Task**.
4. On the page that appears, click the rightward arrow in the middle to show the Actions column in the node list. Find the target node, click **Patch Data** in the Actions column, and then select a mode for generating retroactive data.

   You can also right-click the target node in the directed acyclic graph (DAG), move the pointer over **Run**, and then select a mode for generating retroactive data.

Generate retroactive data for the current node

1. Find the target node, click **Patch Data** in the Actions column, and then select **Current Node**

Retroactively.

2. In the **Patch Data** dialog box, set the parameters.

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.
Data Timestamp	The data timestamp of the retroactive instance.
Node	The name of the node for which you want to generate retroactive data. You cannot change the name.
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ○ If you select **Disable**, only one retroactive instance is generated. The retroactive instance is run multiple times based on the data timestamp in sequence. ○ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated. The retroactive instances are run based on the data timestamp in parallel or in sequence. ■ If the number of days in the data timestamp is smaller than the number of parallel groups, the retroactive instances are run in parallel. For example, the data timestamp is from January 11 to January 13, and you select 4 Parallel Groups. In this case, three retroactive instances are generated for each day in the data timestamp, and the three retroactive instances are run in parallel. ■ If the number of days in the data timestamp is greater than the number of parallel groups, some instances must be run multiple times in sequence while others are run in parallel. For example, the data timestamp is from January 11 to January 13, and you select 2 Parallel Groups. In this case, two retroactive instances are generated. They are run in parallel for once, and one of them must be run for a second time.

3. Click **OK**.

Generate retroactive data for the current and descendant nodes

1. Find the target node, click **Patch Data** in the Actions column, and then select **Current and Descendant Nodes Retroactively.**
2. In the **Patch Data** dialog box, set the parameters, including **Nodes.**

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.

Parameter	Description
Data Timestamp	The data timestamp of the retroactive instance.
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ○ If you select **Disable**, only one retroactive instance is generated. ○ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated.
Nodes	The nodes for which you want to generate retroactive data. You can set **Node Name** and **Node Type** to filter nodes.

3. Click **OK**.

Generate retroactive data for a large number of nodes

1. Find the target node, click **Patch Data** in the Actions column, and then select **Mass Nodes Retroactively**.
2. In the **Patch Data** dialog box, set the parameters. The following table describes the parameters for generating retroactive data for a large number of nodes.

Parameter	Description
Retroactive Instance Name	The name of the retroactive instance. DataWorks automatically generates a name, which can be changed.
Data Timestamp	The data timestamp of the retroactive instance. **Note** We recommend that you do not set this parameter to a long range. Otherwise, the retroactive instance may be delayed due to insufficient resources.
Parallelism	Specifies whether to generate multiple retroactive instances at a time. ○ If you select **Disable**, only one retroactive instance is generated. ○ If you select **2 Parallel Groups**, **3 Parallel Groups**, **4 Parallel Groups**, or **5 Parallel Groups**, multiple retroactive instances are generated.
Nodes	○ If you select the **Current Node** check box, retroactive instances are generated for the current node and its descendant nodes. ○ If you clear the **Current Node** check box, the current node is dry-run and retroactive instances are generated for its descendant nodes.

Parameter	Description
Workspaces	The workspaces of the nodes for which DataWorks needs to generate retroactive data. Select workspaces under **Available Workspaces** and add them to **Selected Workspaces**. Fuzzy match is supported when you search for workspaces under Available Workspaces.
Node Whitelist	The nodes outside the selected workspaces, for which DataWorks needs to generate retroactive data. **Note** You can search for nodes by node ID.
Node Blacklist	The nodes inside the selected workspaces, for which DataWorks does not need to generate retroactive data. **Note** You can search for nodes by node ID.

3. Click **OK**.

Manage retroactive instances in the instance list

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by node name or node ID and set the following parameters to filter instances: **Retroactive Instance Name**, **Node Type**, **Owner**, **Run At**, **Data Timestamp**, **Engine type**, **Engine instance**, **Baseline**, and **My Nodes**. **Note** By default, the data timestamp is set to the previous day of the current day.
DAG	View the DAG of the instance. You can view the running result of the instance in the DAG.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance.
Rerun Descendant Nodes	Rerun the descendant instances of the instance.
Freeze	Pause the scheduling of the instance.
Recover	Resume the scheduling of the instance after it is paused.

Operation	Description
View Lineage	View the lineage of the instance.

Manage retroactive instances in a DAG

Click the name of an instance or DAG in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

> Note After you click Refresh in the upper-right corner, the DAG of the instance is refreshed, but the operational logs are not.

Operation	Description
Show Ancestor Nodes or Show Descendant Nodes	Show ancestor or descendant instances at one or more levels. If a workflow contains three or more instances, the DAG displays only the current instance and hides its ancestor and descendant instances.
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the DataStudio page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
Stop	Stop the instance. You can stop instances only when they are in the Pending (Resource) or Running state. After you perform this operation, the instance enters the Failed state.
Rerun	Rerun the instance if it is in the Failed or an abnormal state.
Rerun Descendant Nodes	Rerun all the descendant instances of the instance.
Set Status to Successful	Set the status of the instance to Successful and run its descendant instances as scheduled. You can perform this operation if an instance fails. > Note This operation applies only to failed instances but not workflows.
Emergency Operations	Perform emergency operations in emergency only. Emergency operations take effect only on the current instance for one time. Select Delete Dependencies to delete dependencies of the current instance. You can perform this operation so that you can start the current instance when the ancestor instances fail and the current instance does not depend on the data of the ancestor instances.

Operation	Description
Freeze	Pause the scheduling of the instance.
Unfreeze	Resume the scheduling of the instance after it is paused.

2.8.3.4. Manage test instances

Test instances are generated when you test auto triggered nodes. You can manage test instances.

Go to the Test Instance page

1. Log on to the DataWorks console.
2. Click  in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Cycle Task > Test Instance**. The Test Instance page appears, where you can view the list and directed acyclic graphs (DAGs) of test instances.

Manage test instances in the instance list

You can manage test instances in the instance list. For example, you can rerun, freeze, or unfreeze instances, set the status of instances to Successful, view the lineage of instances, and check operational logs.

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by node name or node ID and set the following parameters to filter instances: **Node Type**, **Owner**, **Run At**, **Data Timestamp**, **Status**, **Region**, **Engine type**, **Engine instance**, **Baseline**, **My Nodes**, **Tested by Me Today**, and **Frozen**.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. Note This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
More operations	Click **More** in the Actions column and then select an operation. The operations include **Set Status to Successful**, **Freeze**, **Unfreeze**, **View Lineage**, and **View Runtime Log**.

Operation	Description
Batch operations	Select multiple instances and click a button in the lower part of the instance list to perform batch operations. The buttons include **Stop**, **Rerun**, **Set Status to Successful**, **Freeze**, and **Unfreeze**.

Manage test instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the **DataStudio** page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
Stop	Stop the instance. You can stop instances only when they are in the **Pending (Resource)** or **Running** state. After you perform this operation, the instance enters the **Failed** state.
Rerun	Rerun the instance. After the instance is rerun, its descendant instances will be run as scheduled. Perform this operation if an instance fails or it is not run as scheduled. Note This operation applies only to instances in the **Pending (Ancestor)**, **Successful**, or **Failed** state.
Set Status to Successful	Set the status of the instance to **Successful** and run its descendant instances as scheduled. You can perform this operation if an instance fails. Note This operation applies only to failed instances but not workflows.
Freeze	Pause the scheduling of the instance.
Unfreeze	Resume the scheduling of the instance after it is frozen. • If the instance is not run, DataWorks automatically runs this instance after its ancestor instances are run. • If all the ancestor instances have been run, the instance is directly set to Failed. You must manually rerun the instance.

2.8.4. Manually triggered node O&M

2.8.4.1. Manage manually triggered nodes

Manually triggered nodes are nodes whose scheduling type is set to manual before they are committed to the scheduling system.

> Note After a manually triggered node is committed to the scheduling system, it will run only after it is manually triggered.

Manage manually triggered nodes in the node list

The manually triggered node list displays manually triggered nodes that are committed.

Operation	Description
Filter	Find required nodes by specifying filter conditions. You can search for nodes by node name and set the **Owner**, **My Nodes**, and **Modified Today** parameters to filter nodes. > Note When you search for nodes by node name, the search result is affected by other filter conditions you specified. Only nodes that meet both the specified search condition and other filter conditions are returned in the search result.
DAG	Click **DAG** in the Actions column of a node to view the directed acyclic graph (DAG) of the node. You can view the node information, such as the code and lineage, in the DAG.
Run	Click **Run** in the Actions column of a node to run the node. Manually triggered node instances are generated.
View Instances	Click **View Instances** in the Actions column of a node to go to the **Manual Instance** page, where you can view the instances of the node.
More operations	Choose **More** > **Change Owner** in the Actions column of a node to change the owner of the node.
Batch operations	Select multiple nodes and click **Change Owner** in the lower part of the page to change the owner of the nodes.

Manage manually triggered nodes in a DAG

Click the name of a node or **DAG** in the Actions column to view the DAG of the node. In the DAG, you can right-click the node to perform related operations.

Operation	Description
View Code	View the code of the node.
Edit Node	Go to the **DataStudio** page to modify the node.

Operation	Description
View Instances	View the instances of the node.
View Lineage	View the lineage of the node.
Run	Run the node. After you click Run, manually triggered node instances are generated.
Change Resource Group	Change the resource group where the node is run.

2.8.4.2. Manage manually triggered node instances

DataWorks generates manually triggered node instances from manually triggered nodes. Manually triggered nodes do not have node dependencies. They must be run manually.

> **Notice** DataWorks generates alerts only for auto triggered node instances when they fail to be run.

Go to the page for managing manually triggered node instances

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Trigger Task Maintenance > Trigger Instance**. The Trigger Instance page appears, where you can view the list and directed acyclic graphs (DAGs) of manually triggered instances.

Manage manually triggered node instances in the instance list

Operation	Description
Filter	Find required instances by specifying filter conditions. You can search for instances by instance name and set the **Owner**, **Data Timestamp**, and **Run At** parameters to filter instances.
DAG	Click **DAG** in the Actions column to view the DAG of the instance. You can view the running result of the instance in the DAG.
Stop	Click **Stop** to stop the instance if it is in the Running state.
Rerun	Rerun the instance.
Batch stop	Select multiple instances and click **Stop** to stop the selected instances at a time.

Manage manually triggered node instances in a DAG

Click the name of an instance or **DAG** in the Actions column to view the DAG of the instance. In the DAG, you can right-click the instance to perform related operations.

Note A manually triggered node instance does not have dependencies, so only the current instance appears in the DAG.

Operation	Description
View Runtime Log	View the operational logs of the instance if it is in the Running, Successful, or Failed state.
View Code	View the code of the instance.
Edit Node	Go to the DataStudio page to modify the node to which the instance belongs.
View Lineage	View the lineage of the instance.
Stop	Stop the instance.
Rerun	Rerun the instance if it is in the Failed or an abnormal state.

2.8.5. MaxCompute engine O&M

DataWorks Operation Center allows you to view the jobs, quota groups, and projects of MaxCompute.

Prerequisites

A MaxCompute engine is added on the **Project Management** page.

Go to the MaxCompute page

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Engine Maintenance > MaxCompute**. The **Job Queue** tab appears by default.You can view the jobs, quota groups, and projects of MaxCompute.

View jobs

On the **Job Queue** tab, you can view the total number of jobs and numbers of jobs in the **Running**, **Waiting for Resources**, and **Initializing** states.

You can set the **QuotaGroup** and **Project** parameters to filter jobs. You can view the following parameters of each job: **Instance**, **Execution Time**, **CPU Usage (Minimum/Maximum)**, **Memory Usage (Minimum/Maximum)**, **Priority**, **Node**, **Submitted By**, **Project**, **Type**, **Quota Group**, **Cluster**, **Status**, and **Start Time**.

View quota groups

On the **MaxCompute** page, click the **Quotas** tab.

On the **Quotas** tab, you can view the following parameters of each quota group: **Project**, **Quota Group**, **Default**, **Cluster**, **Minimum CPU (cores)**, **Maximum CPU (cores)**, **Minimum Memory (bytes)**, **Maximum Memory (bytes)**, and **Projects**.

You can click the name of a quota group to view the resource usage information about the quota group.

View projects

On the **MaxCompute** page, click the **Projects** tab.

On the **Projects** tab, you can view the following parameters of each project: **Project**, **Owner**, **Cluster**, **Quota Group**, **Storage Used**, **Storage Quota**, **Storage Usage**, and **Files**.

2.8.6. Monitor

2.8.6.1. Overview

The Monitor module is a node monitoring and analysis system of DataWorks. Based on monitoring rules and node running status, the Monitor module determines whether, when, and how to trigger an alert, and whom an alert is sent to. It automatically selects the most appropriate alerting time, notification methods, and recipients.

The Monitor module provides you with the following benefits:

- Improves your efficiency on configuring monitoring rules.
- Prevents invalid alerts from bothering you.
- Automatically covers all important nodes for you.

General monitoring systems cannot meet the requirement of DataWorks. The reasons are as follows:

- DataWorks has numerous nodes, so it is difficult for you to find out the nodes to be monitored. Some DataWorks businesses have a large number of nodes, and dependencies between the nodes are complex. Even if you know the most important node, it is difficult to find all ancestor nodes of the node and monitor them all. In this case, if you simply monitor all nodes, a large number of invalid alerts may be generated. In consequence, you may miss those useful alerts.
- The alerting method varies with nodes. For example, some monitoring tasks require the relevant nodes to run for more than one hour before triggering alerts, while other monitoring tasks require the relevant nodes to run for more than two hours. It is extremely complex to set a monitoring node for each node, and it is difficult to predict the alert threshold value for each node.
- The alerting time varies with nodes. For example, an alert for an unimportant node can be sent after you start working in the morning. An alert for an important node needs to be sent immediately when an error occurs. General monitoring systems cannot tell the importance of each node.
- Different alerts require different operations to turn off.

The Monitor module provides comprehensive monitoring and alerting logic. You only need to provide the node name of your business. Then, the Monitor module automatically monitors the entire process of your node and creates standard alert triggers for the node. In addition, you can customize alerting triggers by completing basic settings.

Currently, the Monitor module has been used for monitoring all important businesses of Alibaba Group. Its full-path monitoring function guarantees the overall data output of all important businesses of Alibaba Group. In addition, it supports analyzing ancestor and descendant node paths to promptly detect risks and provide O&M advice for business departments. These functions of the Monitor module have guaranteed the long-term high stability of businesses in Alibaba Group.

2.8.6.2. Feature description

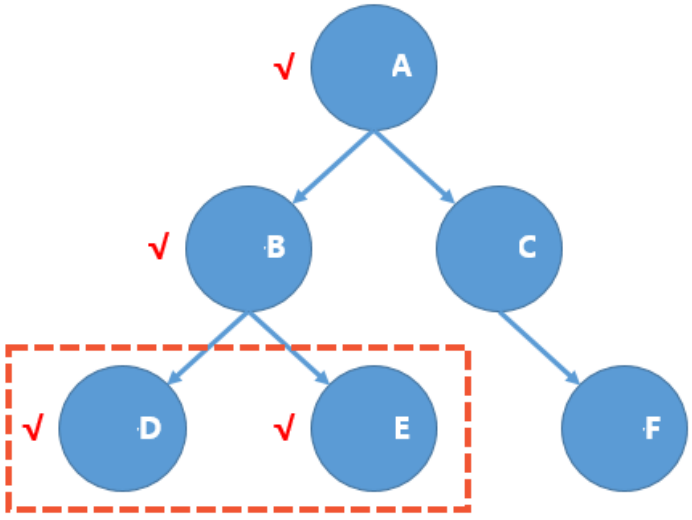
2.8.6.2.1. Baseline alert and event alert

This topic describes the functional logic of baseline alerts and event alerts from the aspects of monitoring scope, node capturing, alert object judgment, alerting time judgment, notification methods, and alert escalation.

Monitoring scope

A baseline is a management unit of a group of nodes, that is, a node group. You can specify nodes to monitor in a baseline.

After a baseline is monitored, all nodes of the baseline and its ancestor nodes are monitored. The Monitor module does not monitor all nodes by default. A node is monitored only when it has descendant nodes that are added to a monitoring baseline. If no descendant nodes are added to a monitoring baseline, the Monitor module does not report any alert even if the node fails.



As shown in the preceding figure, assume that DataWorks has only six nodes, and nodes D and E belong to a monitoring baseline. Nodes D and E and all their ancestor nodes are monitored by the Monitor module. That is, any error or slowdown on node A, B, D, or E will be detected by the Monitor module. However, nodes C and F are not monitored by the Monitor module.

Node capturing

After the nodes to be monitored are specified, if a monitored node incurs an exception, the Monitor module generates an event. All alert decisions are based on the analysis of this event. Two types of node exceptions are available. You can choose **Events** > **Event Type** to view them.

- **Error**: indicates that a node fails to run.
- **Slow**: indicates that the running time of a node is significantly longer than the average

running time of the node in the past periods.

> **Note** If a node times out and then encounters an error, two events are generated.

Alerting time judgment

Buffer, an important concept in the Monitor module, refers to the maximum time period that a node can be delayed. The latest start time of a node is obtained by subtracting the average uptime from the baseline time.

The baseline time of baseline A is 05:00, you must set the latest start time of node E to 04:10. This time is calculated by subtracting the average uptime of node F (20 minutes) and node E (30 minutes) from the baseline time 05:00. This time is also the latest completion time of node B in baseline A.

To ensure that the baseline time of baseline B is 06:00, you must set the latest completion time of node B to 04:00. This time, which is earlier than 04:10, is calculated by subtracting the average uptime of node D (2 hours) from the baseline time 06:00. To meet the baseline time of both baseline A and baseline B, you must set the latest completion time of node B to 04:00.

The latest completion time of node A is 02:00, which is calculated by subtracting the average uptime of node B (2 hours) from 04:00. The latest start time of node A is 01:50, which is calculated by subtracting the average uptime of node A (10 minutes) from 02:00. If node A fails to run before 01:50, it is probable that baseline A is broken.

If node A fails to run at 01:00, its buffer is 50 minutes, which is the difference between 01:00 and 01:50. As demonstrated in this example, buffer reflects the degree of caution for a node exception.

Baseline alert

Baseline alerting is an additional feature developed for baselines that are enabled. Each baseline must provide an alert buffer and committed time. Baseline alerting is the action of notifying the preset alert recipient three times at the interval of 30 minutes when the baseline completion time estimated by the Monitor module exceeds the alert buffer.

Notification method

Currently, baseline alerts are sent to the baseline owner by default. On the **Alert Triggers** page, you can find **Global Baseline Alert Trigger**, click **View Details**, and change the alert trigger method and the alerting action.

Gantt chart function

The Gantt chart function reflects the key path of a node. The function is provided by the **Baseline Instances** module of Monitor.

> **Note** The key path is the slowest upstream link that causes the node to be completed at this time point.

2.8.6.2.2. Custom alert trigger

Alert trigger customization is a lightweight monitoring function of the Monitor module.

You can customize all monitoring alert triggers by setting the following parameters:

- Objects: You can specify nodes, baselines, and workspaces as objects.
- Trigger Condition: Valid values include Completed, Uncompleted, Error, Uncompleted Cycle, and Overtime.
- Notification Method: Valid values include SMS and Email.
- Maximum Alerts: This parameter indicates the maximum number of alert reporting times. If the number of alerting times exceeds the preset threshold, no alerts are generated.
- Minimum Alert Interval: This parameter indicates the minimum time interval at which DataWorks reports alerts.
- Quiet Hours: This parameter indicates the specified period during which no alerts are reported.
- Recipient: This parameter indicates the person who receives alerts. You can set this parameter to the node owner or another recipient.

A monitoring rule uses the following five alert trigger conditions: Completed, Uncompleted, Error, Uncompleted Cycle, and Overtime.

- **Completed**

  A completion alert can be set for nodes, baselines, and workspaces. Once all nodes of the preset objects are completed, the completion alert is reported. If you set a completion alert for a baseline, the alert is reported when all nodes of the baseline are completed.

- **Uncompleted**

  You can set alerts for nodes, baselines, or workspaces that are not completed at a certain time point. For example, if you require that a baseline be completed at 10:00, an alert containing a list of uncompleted nodes is reported once a node in the baseline is not completed at the specified time.

- **Error**

  An error alert can be set for nodes, baselines, and workspaces. Once a node has an error, an alert containing detailed node error information is sent to the recipient.

- **Uncompleted Cycle**

  For the monitoring rules of hourly scheduled nodes, you can separately specify the uncompleted time points in different periods.

- **Overtime**

  An overtime alert can be set for nodes, baselines, and workspaces. Once a monitored node of the preset object is not completed within the specified time, an alert is reported.

2.8.6.3. Instructions

2.8.6.3.1. Baseline instances

On the Baseline Instances page, you can view the information about a baseline.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**. The **Operation Center** page appears.
3. On the Operation Center page, choose **Monitor > Baselines** in the left-side navigation pane. On the **Baselines** page, create a baseline. For more information abut how to create a

baseline, see Create a baseline.

> Note If you have created a baseline, skip this step and directly go to the Baseline Instances page to perform subsequent operations.

4. In the left-side navigation pane, choose Monitor > Baseline Instances. The Baseline Instances page appears. On this page, you can search for baseline instances by condition, such as the data timestamp, owner name or ID, event ID, workspace, and baseline name. You can also click View Details, Handle, and View Gantt Chart in the Actions column to perform corresponding operations on a baseline.

   > Note After creating a baseline, you must enable the baseline so that a baseline instance can be generated.

   A baseline can be in the one of the following four states:

   - Normal: All nodes in the baseline are completed before the alerting time.
   - Alerting: One or more nodes in the baseline are not completed at the alerting time but the committed completion time has not arrived.
   - Overtime: One or more nodes in the baseline are not completed at the committed completion time.
   - Others: All nodes in the baseline are paused or the baseline is not associated with any node.

   You can click View Details, Handle, and View Gantt Chart in the Actions column to perform corresponding operations on a baseline.

   - View Details: Click View Details to go to the Baseline Instance Details page.

     On the Baseline Instance Details page, you can view the general information, critical path, baseline instance information, history graph, and relevant events.

     > Note
     > - In the preceding figure, the data timestamp is `one day before the system time`.
     > - When you create a baseline, you can select By the Day Interval or By the Hour Interval. The Cycle parameter appears as the advanced settings of the Committed Time parameter only when you select By the Hour Interval.

   - Handle: Click Handle to pause the alert generated by the baseline when the baseline is being handled.
   - View Gantt Chart: Click View Gantt Chart to view the critical paths of nodes.

2.8.6.3.2. Baselines

You can create and define baselines on the Baselines page.

Create a baseline

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and

choose **All Products** > **Operation Center**. The **Operation Center** page appears.

3. On the Operation Center page, choose **Monitor** > **Baselines** in the left-side navigation pane.
4. On the **Baselines** page, click **Create Baseline** in the upper-right corner.

> Note Currently, only workspace administrators can create baselines.

5. In the **Create Baseline** dialog box that appears, set the parameters and click **OK**.

Parameter	Description
Baseline Name	The name of the baseline.
Workspace	The workspace of the node associated with the baseline.
Owner	The name or ID of the owner.
Recurrence	Specifies whether the baseline detects nodes by day or hour. ○ **By the Day Interval**: Select this option for nodes scheduled by day. ○ **By the Hour Interval**: Select this option for nodes scheduled by hour.
Nodes	○ **Node**: the node associated with the baseline. Enter the name or ID of a node, and then click the icon on the right to add the node. You can add multiple nodes. ○ **Workflow**: the workflow associated with the baseline. Enter the name or ID of a workflow, and then click the icon on the right to add the workflow. We recommend that you only add the last node of a workflow instead of all nodes.
Priority	The priority of the baseline. A baseline with a higher priority is scheduled first. Currently, the only available priority value is 1.
Estimated Completion Time	The completion time of the node estimated based on the average running time of the node during previous scheduling. If no historical data is available, the message **The completion time cannot be estimated due to a lack of historical data** appears.
Committed Time	The time point when a node should be completed. An alert is triggered if the node is not completed until the time point obtained by subtracting the alert margin threshold from the committed completion time.

Parameter	Description
Margin Threshold	The interval before an alert is triggered. For example, set Committed Time to 3:30 and Margin Threshold to 10 minutes. An alert is triggered if the node is not completed at 3:20. Assume that the average running time of the node is 30 minutes. If the node is not started at 2:50, an alert is triggered. **Note** The average running time of a node can be calculated based on the data of the last 15 days.

6. After a baseline is created, click **Enable** in the Actions column to enable the baseline.

   You can click **View Details**, **Change**, **Enable**, Disable, or **Delete** in the Actions column to perform the corresponding operation on a baseline.

   - **View Details**: Click **View Details** to view the basic information about the baseline.
   - **Change**: Click **Change** to modify the baseline.
   - **Enable** or Disable: Click Enable or Disable to enable or disable the baseline. A baseline instance can be generated only when the corresponding baseline is enabled.
   - **Delete**: Click **Delete** to delete the baseline.

Add a node to a baseline

By default, all nodes in the production environment are in the default baseline of each workspace. When you create a baseline, you actually move nodes from the default baseline to the baseline that you create.

Note A node must belong to a baseline, so you cannot directly remove nodes from the default baseline. Instead, you can create a baseline to move nodes from the default baseline to the new baseline. When you delete a baseline that you create, you actually move the nodes in the baseline to the default baseline.

To change the baseline of a node, perform one of the following operations:

- On the **Baselines** page, click **Create Baseline** in the upper-right corner. Then, create a baseline by following the instructions in the Create a baseline section.
- In the left-side navigation pane, choose Nodes > **Recurring**. On the page that appears, find the node and choose **More** > **Add to Baseline** in the Actions column.

2.8.6.3.3. Events

On the Events page, you can view all events related to slowdown or errors.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products** > **Operation Center**.
3. On the Operation Center page, choose **Monitor** > **Events** in the left-side navigation pane. The **Events** page appears.

You can search for events by condition, such as the event owner, time when an event was detected, event status, event type, and name or ID of a node or node instance.

In the search results, each event is displayed in a row and associated with a node that encounters errors. The worst baseline indicates a baseline with the minimum margin among the baselines affected by an event.

- Click **View Details** in the Actions column of an event. You can view the event occurrence time, alert time, clearance time, historical runtime logs of the node, and detailed node logs.

  You can assign an alert recipient. After you click **View Alerts**, the alert details page corresponding to the event appears. Affected baselines are all descendant baselines affected by the node associated with the event. You can observe descendant baselines and the impact on these baselines and analyze node logs to identify the causes of the event.
- If you click **Handle**, DataWorks records the event handling operation and pauses the alert for the event when the event is being handled.
- If you click **Ignore**, DataWorks keeps the event ignorance record and permanently stops the alert for the event.

2.8.6.3.4. Alert triggers

This topic describes how to customize alert triggers on the Alert Triggers page.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.
3. On the Operation Center page, choose **Monitor > Alert Triggers** in the left-side navigation pane. The **Alert Triggers** page appears.
4. On the Alert Triggers page, click **Create Custom Trigger** in the upper-right corner.
5. In the **Create Custom Trigger** dialog box that appears, set relevant parameters.

Parameter	Description
Trigger Name	The name of the custom alert trigger.
Object Type	The granularity of monitored objects. Valid values: **Node** and **Workflow**.
Objects	The monitored object. Enter the name or ID of a node or workflow and click the icon on the right to add the object.
Trigger Condition	The condition for triggering an alert. Valid values: **Completed**, **Uncompleted**, **Error**, **Uncompleted Cycle**, and **Overtime**.
Maximum Alerts	The maximum number of alert reporting times. If the number of alert reporting times exceeds the preset threshold, no alerts are reported.
Minimum Alert Interval	The minimum time interval at which DataWorks reports an alert.

Parameter	Description
Quiet Hours	The specified period during which no alerts are reported.
Notification Method	The method of reporting alerts. Valid values: **Email** and **SMS**.
Recipient	The person who receives alerts. You can set this parameter to the node owner or another recipient.
DingTalk Chatbot	The DingTalk chatbot to receive alerts.

6. Click **OK** to create the alert trigger.

   On the **Alert Triggers** page, you can click **View Details** in the Actions column of an alert trigger to view the details of the alert trigger.

2.8.6.3.5. Alert information

You can view all alerts on the Alerts page.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.
3. On the Operation Center page, choose **Monitor > Alerts** in the left-side navigation pane. The **Alerts** page appears.

   You can search for alerts by condition, such as the alert trigger ID or name, recipient, alert time, notification method, and alert trigger type.

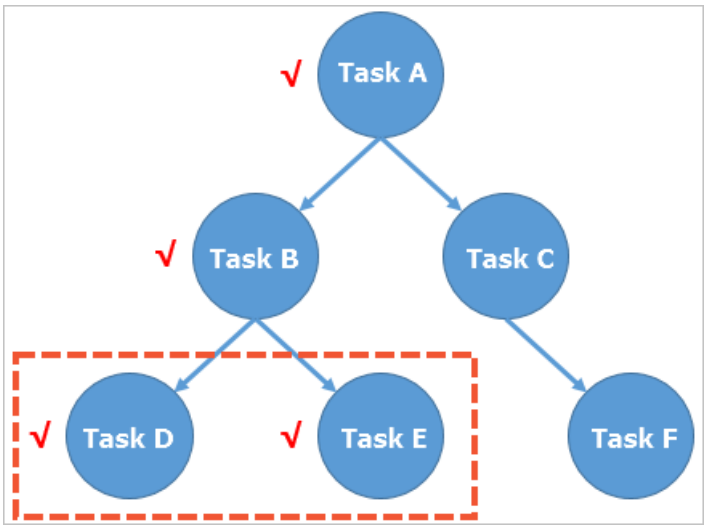
   You can also view alert details such as the notification method and status. To view more alert details, find the target alert and click **View Details** in the Actions column.

2.8.6.4. FAQ

This topic describes the FAQ about the Monitor service.

What can I do if I do not need to receive alerts for a node?

After you create and enable a monitoring baseline, the Monitor service monitors all nodes in the baseline and their ancestor nodes. If a node in the baseline or an ancestor node of the baseline affects data generation of the monitored nodes in the baseline, the Monitor service reports an alert to the node owner.



As shown in the preceding figure, assume that DataWorks has only six nodes, and Nodes D and E belong to a monitoring baseline. The Monitor service monitors Nodes D and E and all their ancestor nodes. Namely, the Monitor service will detect any error or slowdown on Node A, B, D, or E. Nodes C and F are not monitored by the Monitor service.

Nodes A and B are ancestor nodes of Nodes D and E and may affect data generation of the monitored nodes in the baseline. When an error or slowdown occurs on Node A or B, the Monitor service reports an alert to the node owner.

If you do not need to receive alerts for a node, use the following methods:

- If the owners of Nodes D and E do not need to receive alerts, contact the baseline owner to remove Nodes D and E from the baseline.
- If the owner of Node A or B does not need to receive alerts, contact the owners of Nodes D and E to delete the dependency of Nodes D and E on Node A or B.

Why is no alert reported for a baseline in the Overtime state?

Baseline monitoring is controlled by the baseline switch and enabled for nodes. If all nodes are running properly, no alert will be triggered even in the Overtime state. This is because all the nodes are running properly and the Monitor service cannot determine which node has an error. Overtime is a baseline state, indicating that a node is still not completed after the committed time.

If the baseline still enters the Overtime state when all nodes are running properly, consider the following reasons:

- The baseline time is not properly set.
- The node dependency is not properly configured.

Can I disable the Monitor service from reporting an alert for a node that slows down?

The Monitor service reports a node slowdown alert only when a node meets both of the following conditions:

- The node is an ancestor node of an important baseline.
- Compared with its historical performance, the node does slow down.

You can view the descendant baseline affected by the node on the **Event Management** page. Then, you can confirm the impact with the party whose monitoring baseline contains descendant nodes of your node.

- If the node slowdown has a minor impact, you can ignore the alert.
- If the node slowdown has a major impact, maintain your node properly.

Why do I fail to receive an alert for an error node?

The Monitor service reports an alert only for specified nodes when an error occurs. An alert is reported for an error node only when the node meets one of the following conditions:

- The node is an ancestor node of a baseline that has been enabled.
- An alert trigger has been customized.

What can I do if I receive an alert at night?

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click  in the upper-left corner and choose **All Products > Operation Center**.
3. In the left-side navigation pane, choose **Alarm > Event Management**.
4. On the **Event Management** page, disable the event alert. Disable the event alert in one of the following ways:
   - Handle the event that triggers an alert.
     a. Find the target event and click **Handle** in the Actions column.
     b. In the **Handle Event** dialog box, set the **Handling Time** parameter.
     c. Click **OK**.

     > **Note** DataWorks records the event handling operation and pauses alerting for the event when the event is being handled.

   - Ignore the event that triggers an alert.
     a. Find the target event and click **Ignore** in the Actions column.
     b. In the **Ignore Event** message, click **OK**.

     > **Note** DataWorks records the event ignoring operation and permanently stops alerting for the event.

2.9. Security Center

2.9.1. Overview

Security Center provides flexible permission management features. It allows you to request permissions and handle permission requests on the graphical user interface (GUI), and view and manage permissions. Security Center not only improves data security but also facilitates data permission management.

Log on to the DataWorks console. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Security Center**. The **Security Center** page appears.

Security Center consists of the following modules: **My Permissions**, **Authorizations**, and **Approval Center**.

Currently, Security Center provides the following features:

- Self-service permission request: Users can select the required tables to quickly initiate a permission request online. This online request mode is more efficient than the original mode in which users need to contact administrators offline.
- Permission management: Administrators can view the users who have permissions on database tables and revoke permissions as required. Users can also revoke unnecessary permissions themselves.
- Permission request approval: Before granting permissions to users, administrators approve permission requests initiated by users. This implements a visual and process-based permission management system, and supports reviewing the approval process.

In Security Center, you can view permissions on all the tables in an organization, request and revoke table permissions, and approve or reject permission requests.

Each operation in Security Center applies to all the workspaces of a tenant in standard mode and basic mode.

2.9.2. My Permissions

On the My Permissions page, you can view your table and field permissions in a workspace, and request or revoke table and field permissions.

View table and field permissions

1. Move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **My Permissions**. The **Table** tab appears.
2. On this tab, you can select a workspace and specify the environment (for a workspace in standard mode) to view all the tables of the workspace in the specified environment. You can also enter a table name in the search box to search for required tables in fuzzy match mode.

   You can view the names and owners of tables in the workspace, view your permissions on the tables, and request or revoke table and field permissions.

Request table and field permissions

1. Select the tables and fields on which you want to request permissions.
   - Request permissions on a table or some fields in the table

     Select the required fields on which you have no permissions in a table and choose **More > Request Permission** in the Actions column.

Alternatively, choose **More** > **Request Permission** in the Actions column for a table without selecting any fields to request permissions on all the fields in the table.

> **Note** You can request permissions on fields only in a workspace with LabelSecurity enabled. If LabelSecurity is disabled for a workspace, you can request permissions only on tables in this workspace.

- Request permissions on multiple tables and fields

  Select all the required tables and fields and click **Request Permission**.

  > **Note** You can also click **Request Permission** without selecting any tables or fields, and then select the required tables and fields in the **Table Permission Request** dialog box.

2. Set the parameters in the **Table Permission Request** dialog box.

Parameter	Description
Workspace	The name of the workspace, which is automatically entered based on the information you specified on the My Permissions page. You can change the workspace as required.
Environment	The environment of the workspace.
MaxCompute Project	The name of the MaxCompute project.
Grant To	The account for which you request permissions. You can request permissions for the current account or a production account of another workspace you joined.
Reason for Request	The reason why you request permissions.
Objects Requested	The tables on which you request permissions. The tables that you select on the previous page are displayed. You can add tables or delete existing tables as required.

3. After the configuration is completed, click **Submit**. If you do not want to request the permissions, click **Cancel**.

Revoke permissions

You can revoke table and field permissions.

- Revoke field permissions

> Note
> - You can revoke permissions on fields only in a workspace with LabelSecurity enabled.
> - To revoke permissions on all the fields in a table, you can directly revoke the permissions on the table.

i. Choose **More** > **Revoke Field Permission** in the **Actions** column for the table on which you want to revoke permissions.
ii. In the **Revoke Field Permission** dialog box, select the fields on which you want to revoke permissions.
iii. Click **OK**.

- Revoke table permissions
  i. Choose **More** > **Revoke Permission** in the **Actions** column for the table on which you want to revoke permissions.
  ii. In the **Revoke Permission** dialog box, select the permissions you want to revoke.
  iii. Click **OK**.

2.9.3. Authorizations

On the Authorizations page, a workspace administrator can view the accounts that have permissions on tables and fields in each workspace, and revoke unnecessary table and field permissions.

You can move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **Authorizations**. On the **Table** tab that appears, you can view and search for tables in workspaces of the current organization.

On the Table tab, you can select a workspace and specify the environment (for a workspace in standard mode) to view all the tables of the workspace in the specified environment. You can also enter a table name in the search box to search for required tables in fuzzy match mode.

View accounts that have permissions on a table

On the **Table** tab of the **Authorizations** page, click the plus sign (+) in front of a table to view all the accounts that have permissions on the table.

Revoke table permissions

Click **Revoke Permission** in the Actions column for an account to revoke the permissions of the account on the current table.

View field permissions

Click **View Field Permissions** in the Actions column for an account to view the permissions of the account on the fields in the current table.

Revoke field permissions

If LabelSecurity is enabled for the workspace, select fields on the Field Permissions page and click **Revoke Field Permissions** to revoke the permissions on the fields.

2.9.4. Approval Center

On the Approval Center page, you can view your requests and their status, view and handle the requests pending your approval, and view the requests that you have handled.

My Requests

1. Move the pointer over the DataWorks icon in the upper-left corner, and click **Security Center**. In the left-side navigation pane, click **Approval Center**. On the Approval Center page, click the **My Requests** tab.

   On this tab, you can view the information about each of your requests, including **Object Type**, **Workspace**, **Status**, **MaxCompute Project**, **Request Time**, and **Table**.

   > **Note** If a request contains permission requests for tables that belong to different owners, Security Center automatically splits the request into multiple requests by table owner.

2. Click **View** in the Actions column to view the details about a request.

Pending My Approval

1. On the **Approval Center** page, click the **Pending My Approval** tab.

   On this tab, you can view the requests pending your approval. If a request is pending your approval, a red dot appears next to Approval Center and Pending My Approval to remind you.

   You can view the information about each of requests pending your approval, including **Object Type**, **Grant To**, **Request Time**, **Workspace**, **MaxCompute Project**, and **Table**.

2. Click **Handle** in the Actions column to view the details about a request and handle it on the Request Details page. The request details include the progress and objects requested.
3. Enter your comments and click Approve or Reject as required.

Handled by Me

1. On the **Approval Center** page, click the **Handled by Me** tab.

   On this tab, you can view the information about each request that you have handled, including **Object Type**, **Grant To**, **Result**, **Workspace**, **MaxCompute Project**, **Table**, and **Request Time**.

2. Click **View** in the Actions column to view the details about a request. The request details include the progress and objects requested.

2.9.5. FAQ

This topic describes the frequently asked questions (FAQs) about the Security Center service of DataWorks.

- Q: What permissions can I request in Security Center?

  A: In Security Center, you can request permissions on tables in DataWorks workspaces in the development environment and production environment.

- Q: What is the relationship between Data Management and Security Center?

  A: Security Center is a product that upgrades and replaces the permission and security features in Data Management. You can choose **Security Center** > **My Permissions** to view the permissions requested or granted by using the `odpscmd grant` command in **Data Management**.

  If you want to request other permissions and handle permission requests on the GUI, go to **Security Center** and perform operations as required. The **Data Management** service does not support permission request and approval any more.

- Q: Why cannot I select fields when I request permissions?

  A: If LabelSecurity is enabled for a workspace, you can request permissions on fields in this workspace. If LabelSecurity is disabled for a workspace, you can request permissions only on tables in this workspace.

- Q: Who will handle my request?

  A: Your request is handled by a workspace administrator or a table owner. After either of them approves or rejects your request, the request is closed.

- Q: Why do I find two requests on the **My Requests** page after I submit only one request?

  A: The tables in your request belong to two owners. In this case, Security Center automatically splits your request into two by table owner.

- Q: I request permissions on a field for one month only. Why does the validity period of the permissions become permanent after my request is approved?

  A: The security level of this field is zero or not higher than the security level of your account.

- Q: Why do I obtain permissions on some tables and fields on which I have not requested any permissions?

  A: The possible causes are as follows:

  - An administrator has granted the permissions to you by running commands in the DataWorks console.
  - After your request is approved in Security Center, Security Center also grants you the permissions on fields whose security level is zero or not higher than the security level of your account, even though you have not requested the permissions.

- Q: Why does a request disappear from the **Pending My Approval** tab before I handle it?

  A: Another workspace administrator or the table owner has approved or rejected the request. The request is closed and no longer needs to be handled.

- Q: What can I do if the message "An error occurred in the MaxCompute project" appears when I specify the workspace and environment?

  A: Send the error message and error code to a workspace administrator for troubleshooting.

- Q: Why do I fail to revoke permissions on a field?

  A: You can revoke permissions only on the fields whose security level is higher than the security level of your account.

- Q: Why do I fail to request permissions by using my tenant account?

  A: By default, a tenant account has all permissions. Therefore, you do not need to request permissions for your tenant account. The tenant account hides unnecessary operations such as permission request. This does not affect the use of the tenant account.

- Q: In **Security Center**, can I view the permission request and approval records of **Data Management**?

  A: Security Center and Data Management have not synchronized permission request and approval records yet. You need to go to **Data Management** to view the permission request and approval records of Data Management.

- Q: Can I revoke permissions based on the request records in Security Center?

  A: Currently, Security Center is not the only service that provides authorization. To facilitate permission revocation, the Authorizations page in Security Center provides an access control list (ACL) of all users, regardless of the authorization channel. You can revoke any granted permissions without using the request records.

- Q: A permission request submitted in **Data Management** has not been approved yet. Do I need to submit it again in Security Center?

  A: Security Center and Data Management have not synchronized permission request and approval records yet. You need to submit the permission request again in Security Center.

- Q: How do I specify the LabelSecurity parameter for fields?

  A: You need to go to **Data Map** to set the LabelSecurity parameter for fields.

2.10. Data Quality

2.10.1. Overview

DataWorks provides a Data Quality service for you to control the data quality of disparate connections. In Data Quality, you can check data quality, configure alert notifications, and manage connections.

Relying on DataWorks, Data Quality provides a comprehensive data quality solution that has various features. For example, you can detect data, compare data, monitor data quality, and use intelligent alerting.

Data Quality monitors data in datasets. Currently, it allows you to monitor MaxCompute tables and DataHub topics. When offline MaxCompute data changes, Data Quality checks data and blocks nodes if it detects exceptions. This prevents nodes from being affected. In addition, Data Quality allows you to manage the check result history so that you can analyze and evaluate the data quality.

For streaming data, Data Quality uses DataHub to monitor data streams and sends alert notifications to subscribers if it detects stream discontinuity. You can also set the alert severity such as warning and error alerts, and the alert frequency to minimize repeated alerts.

The following figure shows the monitoring flowchart in Data Quality.

> **Note** Data Quality monitors the quality of data from MaxCompute and DataHub datasets. To use Data Quality features, you need to create tables and write data to the tables.
>
> You can create MaxCompute tables and write data to the tables in the MaxCompute console or in the DataWorks console.

Log on to the DataWorks console. Click the DataWorks icon in the upper-left corner and choose **All Products** > **Data Quality** to go to the **Data Quality** page.

2.10.2. Features

2.10.2.1. Dashboard

As the homepage of Data Quality, the Dashboard page displays an overview of alerts and blocks for subscribed nodes. You can set filter conditions to view the required alerts and blocks.

Card	Description
My MaxCompute Partition Subscriptions	Displays the number of MaxCompute partitions with alerts or blocks and the number of normal MaxCompute partitions on the current day. You can click this card to go to the Search by Node page for the MaxCompute connection and view alert details.
My DataHub Topic Subscriptions	Displays the number of DataHub topics with alerts and the number of normal DataHub topics on the current day. You can click this card to go to the Search by Node page for the DataHub connection and view alert details.
Current task alarm condition	Displays alerts for MaxCompute and DataHub connections of the current workspace on the current day.
Current task blocking situation	Displays blocks for the MaxCompute connection of the current workspace on the current day.
Task Alarm Situation Trend	Displays the trend chart of alerts for MaxCompute and DataHub connections. You can view the alert trend in the past 7 or 30 days, or a custom time period within the past three months.
Task Blocking Situation Trend Graph	Displays the trend chart of blocks for MaxCompute and DataHub connections. You can view the block trend in the past 7 or 30 days, or a custom time period within the past three months.

2.10.2.2. My Subscriptions

The My Subscriptions page displays all nodes subscribed by your account.

Go to the My Subscriptions page

Currently, Data Quality allows you to monitor MaxCompute tables and DataHub topics. You can select a connection on the **My Subscriptions** page and search for subscribed nodes of the connection.

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
3. In the left-side navigation pane, click **My Subscriptions**. The **My Subscriptions** page appears.

Subscribed MaxCompute connections

On the **My Subscriptions** page, select **MaxCompute** from the connection drop-down list in the upper-left corner. All the subscribed MaxCompute connections appear.

- You can click a partition expression on the right to go to the **Rules** page. For more information,

see MaxCompute monitoring.

- You can click **View Check Results** in the Actions column for a partition expression to go to the Search by Node page.
- Data Quality supports the following four notification methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.
- You can click **Cancel Subscription** to unsubscribe from the connection.

Subscribed DataHub connections

On the **My Subscriptions** page, select **DataHub** from the connection drop-down list in the upper-left corner. All the subscribed DataHub connections appear.

- After you click **Alerts** for a topic, the **Alerts** page appears, allowing you to view detailed information about the rule alert.
- You can click **Notification Method** for a topic to change the notification method of the rule alert.
- You can click **Cancel Subscription** in the Actions column for a topic to unsubscribe from the topic.

2.10.2.3. Configure monitoring rules

Data Quality can monitor data in the MaxCompute, DataHub, and E-MapReduce data stores. This topic describes how to configure a rule for monitoring a table or topic.

Go to the Monitoring Rules page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
3. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane. On the Monitoring Rules page, you can select ODPS, Datahub, or EMR from the Engine/Data Source drop-down list.
   - If you select ODPS or EMR, all tables in the current MaxCompute or E-MapReduce data store appear. You can also switch to another data store or enter a keyword in the search box to search for topics or tables.
   - If you select Datahub, all topics and dimension tables in the current DataHub data store appear. You can also switch to another data store or enter a keyword in the search box to search for topics or tables.
4. Find the target table or topic and click **View Monitoring Rules** in the Actions column. The rule configuration page for the table or topic appears.

   Data Quality allows you to configure template rules and custom rules for a table or topic.

   > **Note** Before you configure a template rule for a table, you must configure a partition filter expression.

Create a template rule

1. Click **View Monitoring Rules** in the Actions column of a table or topic.

2. On the rule configuration page that appears, click the partition filter expression for which you want to configure a template rule. Then, click **Create rules**. In the Create rules right-side pane, the **Template Rules** tab appears.

   On the **Template Rules** tab, click **Add Monitoring Rule** or **Quick Create** to create a template rule.

   - **Click Add Monitoring Rule.**

     Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Rule Type	The type of the rule. Valid values: Rule Type and Soft. ■ If you select Rule Type, error alerts are reported and descendant nodes are blocked, whereas warning alerts are reported but descendant nodes are not blocked. ■ If you select Soft, error alerts are reported but descendant nodes are not blocked, whereas warning alerts are not reported and descendant nodes are not blocked.
Field	The fields to be monitored. You can select **All Fields in Table** or select a field of a numeric type or non-numeric type.
Template	The template to apply to the rule. Data Quality supports 37 rule templates. **Note** You can set field-specific rules of the average value, accumulated value, minimum value, and maximum value only for numeric fields.
Comparison Method	The comparison method of the rule. Valid values: **Absolute Value**, **Raise**, and **Drop**.

Parameter	Description
Thresholds	■ You can calculate the fluctuation by using the following formula: Fluctuation = (Sample - Baseline)/Baseline ■ You can calculate the fluctuation variance only for numeric fields such as BIGINT and DOUBLE fields by using the following formula: Fluctuation variance = (Sample - Baseline)/Standard deviation Note The sample and baseline are defined in the following way: ■ Sample: the sample value for the current day. For example, if you need to check the fluctuation of table rows on an SQL node in a day, the sample is the number of table rows on the current day. ■ Baseline: the comparison value from the previous N days. Examples: ■ If you need to check the fluctuation of table rows on an SQL node in a day, the baseline is the number of table rows on the previous day. ■ If you need to check the fluctuation of the average number of table rows on an SQL node in seven days, the baseline is the average number of table rows in the last seven days. You can set **Warning Threshold** and **Error Threshold** to monitor data at different severities: ■ If the fluctuation does not exceed the warning threshold, Data Quality determines that data is normal. ■ If the fluctuation exceeds the warning threshold but does not exceed the error threshold, Data Quality reports a warning alert. ■ If the fluctuation exceeds the error threshold, Data Quality reports an error alert. ■ If you do not specify the warning threshold, Data Quality reports error alerts or normal based on the monitoring result. ■ If you do not specify the error threshold, Data Quality reports warning alerts or normal based on the monitoring result. ■ If you specify neither the warning threshold nor the error threshold, Data Quality reports error alerts if it detects anomalies. However, you must specify at least one of the two thresholds. If you specify neither of them, Data Quality applies default values, namely, 10% for the warning threshold and 50% for the error threshold.

- **Click Quick Create.**

Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Field	The fields to be monitored. You can select **All Fields in Table** or a specific field of a numeric type or non-numeric type.
Trigger	The trigger condition of the rule. If you select All Fields in Table for the Field parameter, **The number of rows is greater than 0** is selected by default.

3. Click **Batch Create**.

Create a custom rule

If template rules do not meet your requirements for monitoring the data quality, you can create custom rules.

1. Click **View Monitoring Rules** in the Actions column of a table or topic.
2. On the rule configuration page that appears, click the partition filter expression for which you want to configure a custom rule. Then, click **Create rules**. In the Create rules right-side pane, the **Template Rules** tab appears.
3. Click the **Custom Rules** tab. On the **Custom Rules** tab, click **Add Monitoring Rule** or **Quick Create** to create a custom rule.
   - **Click Add Monitoring Rule.**

     Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Field	The fields to be monitored. You can select **All Fields in Table**, **SQL Statement**, or a specific field. ■ If you select All Fields in Table or a specific field, you can specify the WHERE clause to customize filter conditions based on business requirements. ■ If you select SQL Statement, you can customize the SQL logic to set a rule. The return value is the value in a row of a column.

Parameter	Description
Rule Type	The type of the rule. Valid values: Rule Type and Soft. ■ If you select Rule Type, error alerts are reported and descendant nodes are blocked, whereas warning alerts are reported but descendant nodes are not blocked. ■ If you select Soft, error alerts are reported but descendant nodes are not blocked, whereas warning alerts are not reported and descendant nodes are not blocked.
Sampling Method	The statistical function. Valid values: **count** and **count/table_count**.
Filter	The filter condition of the rule. For example, if you need to query partitions of the table based on a specific data timestamp, you can specify `pt=$[yyyymmdd-1]` as the filter condition.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.
Comparison Method	The comparison method of the rule. If you set Check type to Numeric type, the values that are optional for this parameter include **Greater Than**, **Greater Than or Equal To**, **Equal To**, **Unequal To**, **Less Than**, and **Less Than or Equal To**.
Verification Method	The verification method of the rule. If you set Check type to Numeric type, you can only set this parameter to **Compare with a specified value**.
Expected Value	The expected value of the rule.
Description	The description of the rule.

- **Click Quick Create.**

  Set parameters in the rule configuration section that appears, as described in the following table.

Parameter	Description
Rule Name	The name of the rule.
Trigger	The type of the rule. You can select only **Values Duplicated in Multiple Fields**.
Field	The fields to be monitored.

Associate a custom node with Data Quality monitoring rules

Before you associate a custom node with Data Quality monitoring rules, make sure that the custom node is created and committed to the production environment. For more information, see Create a custom node type.

You can use one of the following methods to associate a custom node with Data Quality monitoring rules:

- Associate a custom node with Data Quality monitoring rules on the Data Quality page.
  i. Log on to the DataWorks console.
  ii. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Quality**.
  iii. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
  iv. Select the target data store from the Engine/Data Source drop-down list, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.
  v. On the rule configuration page that appears, click the partition filter expression for which monitoring rules are configured.
  vi. Click **Manage Linked Nodes**.
  vii. In the **Manage Linked Nodes** dialog box, select the target workspace, enter the ID or name of the custom node, and then click **Create**.
- Associate a custom node with Data Quality monitoring rules on the Operation Center page.
  i. Log on to the DataWorks console.
  ii. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Operation Center**.
  iii. In the left-side navigation pane, choose **Cycle Task Maintenance > Cycle Task**.
  iv. Find the target node and choose **More > Configure Data Quality Rules** in the Actions column.
  v. In the **Configure Data Quality Rules** dialog box, set the **Workspace**, **Table Name**, **Engine type**, **Engine instance**, and **Partition Expression** parameters, and click **Add**.

2.10.2.4. View monitoring results

The Node Query page displays the monitoring results of rules. After monitoring rules are triggered, you can go to the Node Query page to view the monitoring results of the rules.

Go to the Node Query page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click  in the upper-left corner and choose **All Products > DataAnalysis**.
3. On the Data Quality page, click **Node Query** in the left-side navigation pane.On the **Node Query** page, you can set parameters, such as the **Engine/Data Source**, **Status**, and **My Subscriptions** parameters, to filter nodes and view the monitoring results.

View the monitoring results of E-MapReduce and MaxCompute tables

GUI element	Description
Engine/Data Source	The name of the compute engine. In this example, select **EMR** or **ODPS**.

GUI element	Description
Engine/Database Instance	The E-MapReduce instance or MaxCompute project where the desired tables reside.
Status	The monitoring result of rules. Pay attention to partitions that trigger alerts or blocks.
Data Timestamp	The data timestamp.
My Subscriptions	Specifies whether to display only monitoring results of tables that you subscribed to.
Run At	The time when rules were triggered.
Table Name	The name of the table whose monitoring results you want to view.
Node	The node that triggered rules.
Details	Click **Details** in the **Actions** column of a table. On the page that appears, you can perform the following operations on each rule configured for the table: • Click **View History Check Results** in the Actions column of a rule to view the monitoring result history of the rule. • Enter comments on a rule based on the execution status of the rule. Perform the following steps to enter comments on a rule: i. Click **Problem Handling** in the Actions column of the rule. ii. In the **Problem Handling** dialog box, set the **Handling Method** and **Comments** parameters. iii. Click **OK**. **Notice** You can only use the problem handling feature in DataWorks Enterprise Edition or higher. • Click **Handling Logs** in the Actions column of a rule to view the processing history of the rule.
Rules	Click **Rules** in the Actions column of a table to go to the rule configuration page for the table. On this page, you can view partition filter expressions and rules configured for the table, and modify the rules as required. For more information, see MaxCompute monitoring.
View Log	Click **View Log** in the Actions column of a table to view the operational logs of rules configured for the table.
View Statistics	Click **View Statistics** in the Actions column of a table to view rule execution information about the table, including the number of rows and the table size.

View the monitoring results of DataHub topics

GUI element	Description
Engine/Data Source	The name of the compute engine. In this example, select Datahub.
Configure a data source	The name of the DataHub connection.
Status	The monitoring result of rules. Pay attention to topics that trigger alerts or blocks.
Topic	The name of the topic whose monitoring results you want to view.
My Subscriptions	Specifies whether to display only monitoring results of topics that you subscribed to.
Clear	Click Clear to clear the filter conditions you specified.
View Log	Click View Log in the Actions column of a topic to view the operational logs of rules configured for the topic.
Alerts	Click Alerts in the Actions column of a topic. On the Alerts page, you can view details about alerts triggered by the topic. On the Alerts page, you can click Close in the Actions column of an alert. In the message that appears, click OK to disable the alert.

2.10.2.5. Report Template Management

On the Report Template Management page, you can create a template of data quality reports. DataWorks can periodically generate and send data quality reports based on the template.

Create a report template

1. Log on to the DataWorks console. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose All Products > Data Quality.
2. On the Data Quality page that appears, choose Configuration > Report Template Management in the left-side navigation pane. The Report Template Management page appears.
3. Click Create Report Template. On the Create Report Template page that appears, set required parameters.

Section	Parameter	Description
	Name	The name of the report template.

Section	Parameter	Description
Basic settings	**Sending Cycle**	The interval at which reports are sent. Valid values: **Every Day**, **Every Week**, **Every Month**, and **Do Not Send**. If you set Sending Cycle to Every Week or Every Month, you also need to specify the specific day on which reports are sent.
	Timespan	The number of days before the current day. DataWorks generates reports based on the data of those days. The maximum value of this parameter is 30.
Statistics of Rule Configuration The Statistics of Rule Configuration section displays metrics about rule configuration for offline data and real-time data. You can select metrics based on your needs.	**Offline data**	The metrics about rule configuration for tables in the workspace. The metrics include **Table count**, **Partition expression count**, **Count of rule on offline data**, and **Rule coverage on tables**. The Rule coverage on tables metric indicates the ratio of tables for which quality monitoring rules are configured.
	Realtime data	The metrics about rule configuration for topics in the workspace. The metrics include **Topic count**, **Count of rule on realtime data**, **Count of rule on cut off data**, **Rule coverage on topic**, **Count of rule on delayed data**, and **Count of customized rule**. The Rule coverage on topic metric indicates the ratio of topics for which quality monitoring rules are configured.

Section	Parameter	Description
Statistics of Rule Execution The Statistics of Rule Execution section displays metrics about rule running for offline data and real-time data. You can select metrics based on your needs. Quality reports display the selected metrics in charts.	**Offline data**	The metrics about rule running for tables in the workspace. The metrics are classified into the following types: **About rules**, **About partitions**, and **About tables**.
	Realtime data	The metrics about rule running for topics in the workspace. The metrics are classified into the following types: **About messages**, **About alarms**, and **About cut-offs**.
Manage Subscriptions	**Subscription Method**	The method used to notify report subscribers of new reports. Currently, DataWorks sends emails to notify report subscribers of new reports.
	Recipient	The recipient of report notifications. You can add multiple recipients.
	Actions	The operations that you can perform on the subscription. You can click **Save** or **Cancel** in the Actions column of a subscription to save or cancel the subscription.
	Add Subscription	The button that allows you to add a subscription.

4. Click **Save** in the upper-right corner. A template of data quality reports is generated.

Preview a report template

After creating a report template, you can click **Preview** in the upper-right corner of the Create Report Template page to view the display format of reports generated based on this template.

> **Note** If report subscribers view reports through email notifications, they can only view the reports in tables. If they view reports on the Data Quality page, they can view reports in tables or charts.

2.10.2.6. Manage rule templates

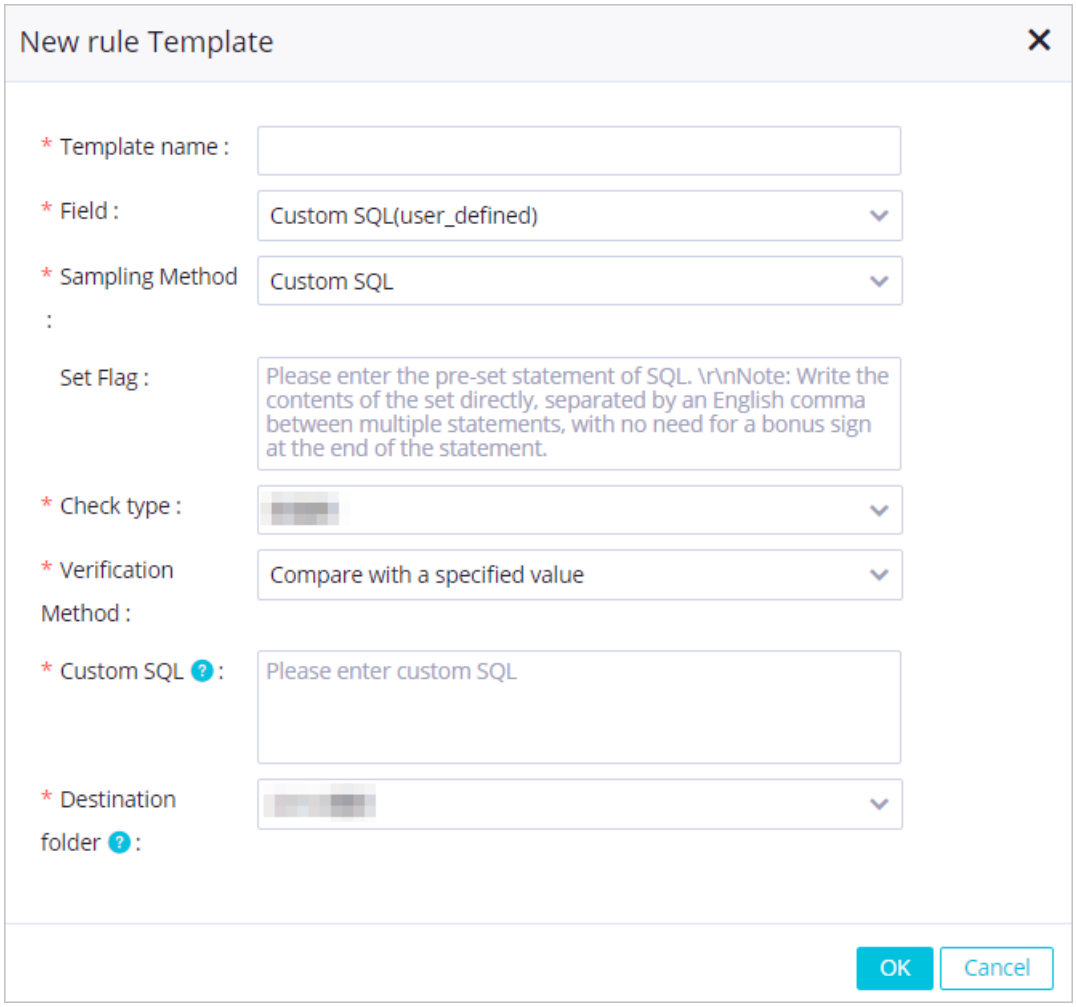
In Data Quality, you can manage a set of custom rule templates and use the rule templates to improve the efficiency of rule configuration.

Context

You can create a rule template on the **Rule Templates** and **Monitoring Rules** pages. After the rule template is created, you can manage and use it.

Create a rule template on the Rule Template Management page

1. Log on to the DataWorks console.
2. On the DataStudio page, click in the upper-left corner and choose **All Products > Data Quality**.
3. On the Data Quality page, choose **Configuration > Rule Templates** in the left-side navigation pane.
4. On the Rule Templates page, click and select **Create Folder**.
5. In the **Create Folder** dialog box, set the **Name** and **Location** parameters and click **OK**.
6. Right-click the folder name and select **Create Rule Template**. You can also rename or delete a folder.
7. In the **Create Rule Template** dialog box, set relevant parameters.



Parameter	Description
Template Name	The name of the rule template.
Field	The fields to be monitored and the statistical function. You can only set the two parameters to **Custom SQL**.
Sampling Method	
Set Flag	The `SET` clause of the SQL statement for querying the field to be monitored. **Note** Separate multiple statements with commas (,). You do not need to add a semicolon (;) at the end of each statement.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.
Verification Method	The verification method of the rule template. The verification methods that can be selected vary with the threshold type. ○ If you set the **Check type** parameter to **Numeric type**, you can only set this parameter to **Compare with a specified value**. ○ If you set the **Check type** parameter to **Fluctuation**, the values that are optional for this parameter include **Compare the current value with the average value of the last 7 days**, **Compare the current value with the average value of the last 30 days**, **Compare the current value with the value 1 day before**, **Compare the current value with the value 7 days before**, **Compare the current value with the value 30 days before**, **The variance between the current value and the value 7 days before**, **The variance between the current value and the value 30 days before**, **Compare with the value 1, 7, and 30 days before** and **Compare with the value of the previous cycle**.
Custom SQL	The custom SQL statement. You can use ${tableName} as the table name.
Location	The name of the folder to which you want to store the custom rule template.

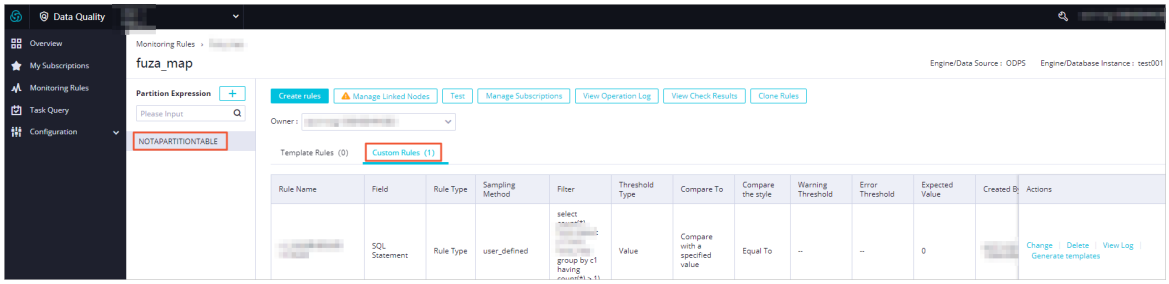
8. Click **OK**.

Create a rule template on the Monitoring Rules page

1. Go to the **Data Quality** page.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. On the Monitoring Rules page, select the compute engine or data store, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.

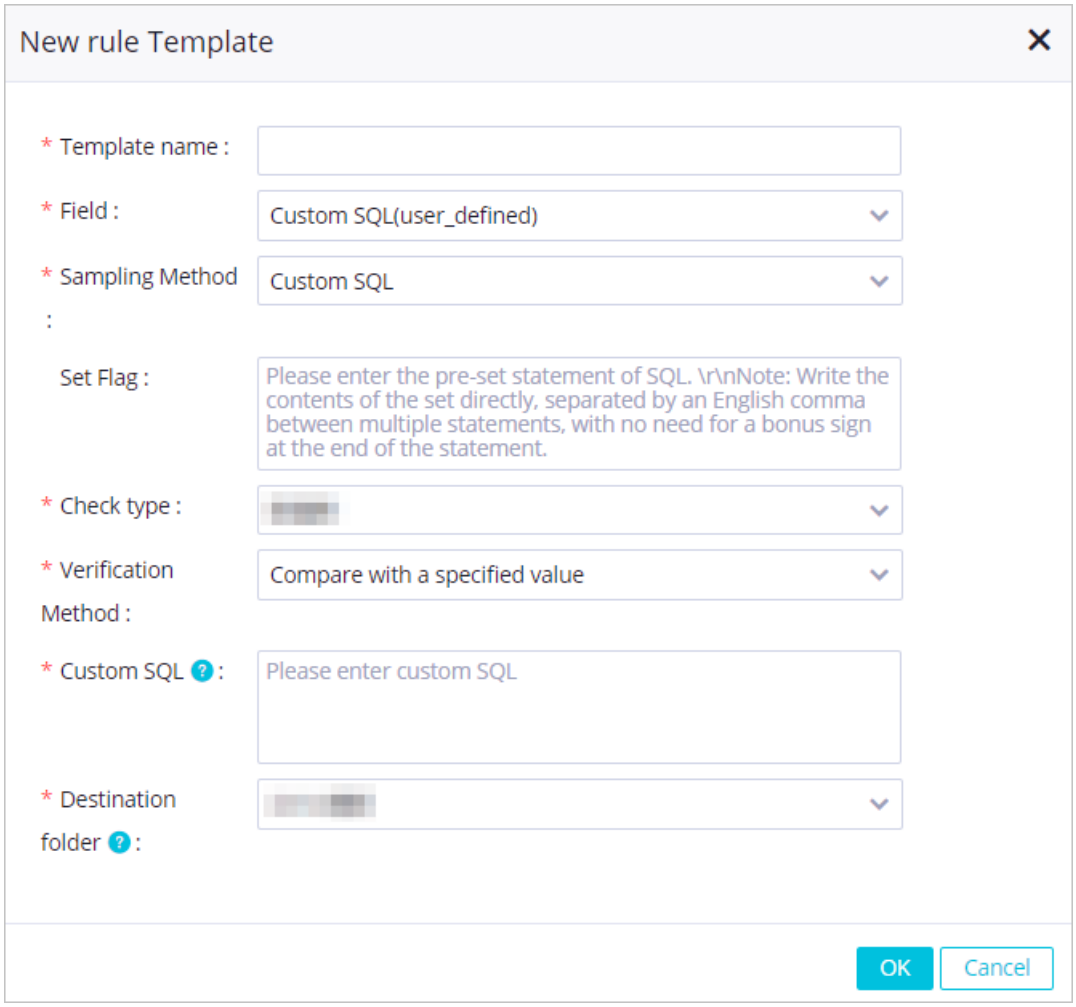
   Note This topic uses a MaxCompute table as an example.

4. Click a partition filter expression and then click the **Custom Rules** tab.



Note For more information about how to create custom rules, see **Custom rules**.

5. On the **Custom Rules** tab, find the target custom rule and click **Generate Template** in the **Actions** column.
6. In the **Create Template** dialog box, set relevant parameters.



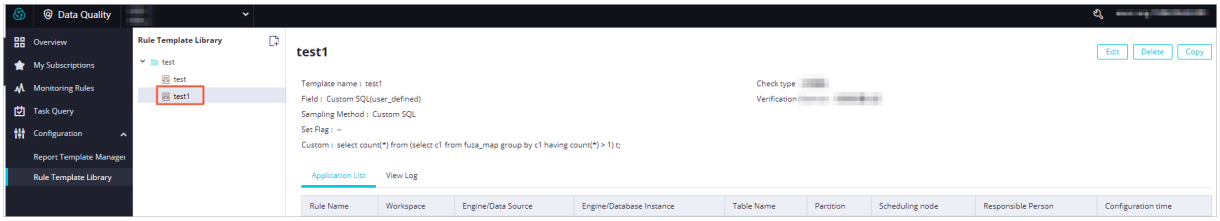
Parameter	Description
Template Name	The name of the rule template.
Field	The fields to be monitored and the statistical function. You can only set the two parameters to **Custom SQL**.
Sampling Method	

Parameter	Description
Set Flag	The `SET` clause of the SQL statement for querying the field to be monitored. **Note** Separate multiple statements with commas (,). You do not need to add a semicolon (;) at the end of each statement.
Check type	The threshold type of the rule. Valid values: **Numeric type** and **Fluctuation**.
Verification Method	The verification method of the rule template. The verification methods that can be selected vary with the threshold type. ○ If you set the **Check type** parameter to **Numeric type**, you can only set this parameter to **Compare with a specified value**. ○ If you set the **Check type** parameter to **Fluctuation**, the values that are optional for this parameter include **Compare the current value with the average value of the last 7 days**, **Compare the current value with the average value of the last 30 days**, **Compare the current value with the value 1 day before**, **Compare the current value with the value 7 days before**, **Compare the current value with the value 30 days before**, **The variance between the current value and the value 7 days before**, **The variance between the current value and the value 30 days before**, **Compare with the value 1, 7, and 30 days before** and **Compare with the value of the previous cycle**.
Custom SQL	The custom SQL statement. You can use ${tableName} as the table name.
Location	The name of the folder to which you want to store the custom rule template.

7. Click **OK**.
8. In the left-side navigation pane, choose **Configuration** > **Rule Templates** to view the created rule template.

Manage an existing rule template

On the Rule Templates page, you can click the name of a rule template to go to the template details page. On this page, you can view, edit, delete, or copy the rule template.



Action	Description

Action	Description
View	You can view the parameter configuration, the rules that use the rule template, and logs of the rule template: • The **Application List** tab displays the rules that use the rule template. • The **View Log** tab displays the logs of operations performed on the rule template, including the user who performed each operation, the time when each operation was performed, and the operation details.
Edit	Click **Edit** in the upper-right corner. In the **Edit Rule Template** dialog box, modify the required parameters, and click **OK**.
Delete	Click **Delete** in the upper-right corner. In the **Delete Template** message, click **OK**.
Copy	Click **Copy** in the upper-right corner. In the **Copy Rule Template** dialog box, set the **Template Name** and **Location** parameters and click **OK**.

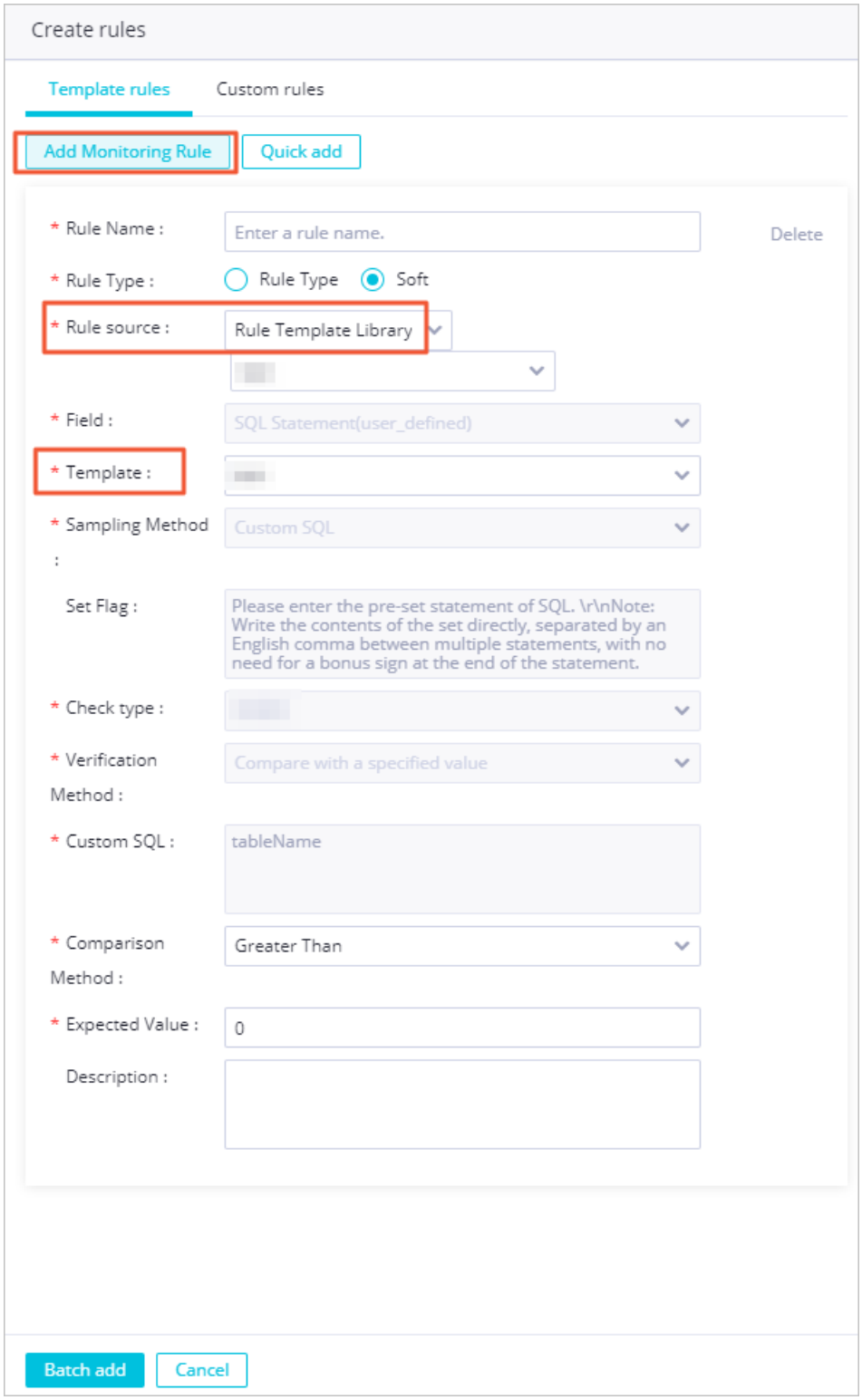
Use a rule template

When you create a monitoring rule, you can select a custom rule template to create the rule based on the rule template.

1. Go to the **Data Quality** page.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. On the Monitoring Rules page, select the compute engine or data store, find the target table or topic, and then click **View Monitoring Rules** in the Actions column.

   > **Note** This topic uses a MaxCompute table as an example.

4. Click a partition filter expression and click the **Custom Rules** tab.
5. On the **Template Rules** tab of the **Create rules** right-side pane, click **Add Monitoring Rule**.
6. Set the parameters for the rule. Specifically, set the **Rule Source** parameter to **Rule Templates** and select a rule template. For more information about the parameter description, see Rules.



7. Click **Batch Create**.

2.10.3. User guide

2.10.3.1. Configure monitoring rules for MaxCompute

The Monitoring Rules page is the most important part of Data Quality, where you can configure rules to monitor data in E-MapReduce, MaxCompute, and DataHub. This topic describes how to configure monitoring rules for MaxCompute.

Add a MaxCompute connection

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration**.
3. On the Data Integration page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
4. Click **Add Connection** in the upper-right corner to add a MaxCompute connection.

Select the MaxCompute connection

1. On the current page, click  in the upper-left corner and choose **All Products > Data Quality**.
2. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
3. Select **ODPS** from the **Engine/Data Source** drop-down list to display all tables in the MaxCompute data store.You can search for a table by table name. Fuzzy search based on the initial letters of a table name is supported.
4. Find the target table and click **View Monitoring Rules** in the Actions column.

Configure a partition filter expression

In Data Quality, you must configure rules based on a partition filter expression:

- To configure rules for a non-partitioned table, you can specify NOTAPARTITIONTABLE as the partition filter expression.
- To configure rules for a partitioned table, you can specify a data timestamp expression, such as $[yyyymmdd], or a regular expression as the partition filter expression.

On the rule configuration page of a table, click + next to **Partition Expression** to add a partition filter expression.

You can create a partition filter expression or select a recommended partition filter expression.

- Create a partition filter expression

  In the **Add Partition** dialog box, enter a partition filter expression that conforms to the syntax as required. For a non-partitioned table, select **NOTAPARTITIONTABLE** from the recommended partition filter expressions.

  - For a table with only one partition, follow the format: Partition key=Partition value. The partition value can be either a constant or a system parameter. You must configure partition expressions by using the last partition.
  - For a table with multiple partitions, follow the format: Partition key 1\=Partition value/Partition key 2=Partition value/Partition key N=Partition value. Each partition value can be either a constant or a system parameter. You must use brackets [ ] to indicate a parameter, such as $[yyyymmdd-N].

The data timestamp configured in a partition filter expression also determines the recurrence of the partition filter expression. For example, if the data timestamp is the date of five days ago, the partition filter expression is triggered every five days. The following table describes supported partition filter expressions.

Partition filter expression	Description
dt=$[yyyymmdd-N]	Indicates N days before.
dt=$[yyyymm01-1]	Indicates the first day of each month.
dt=$[yyyymm01-Nm]	Indicates the first day of the month that is N months before the current month.
dt=$[yyyymmld-1]	Indicates the last day of each month.
dt=$[yyyymmld-1m]	Indicates the last day of the month that is N months before the current month.
dt=$[hh24miss-1/24]	Indicates one hour before the hour specified by the data timestamp.
dt=$[hh24miss-30/24/60]	Indicates half an hour before the hour specified by the data timestamp.
$[yyyymmdd]	Indicates the data timestamp.
$[yyyymmdd-1]	Indicates one day before the data timestamp of the current instance.
$[yyyymmddhh24miss]	Indicates the data timestamp of the current instance. Follow the `yyyymmddhh24miss` format by understanding the following format description: ◦ yyyy indicates a four-digit year. ◦ mm indicates a two-digit month. ◦ dd indicates a two-digit day. ◦ hh24 indicates a two-digit hour (24-hour clock). ◦ mi indicates two-digit minutes. ◦ ss indicates two-digit seconds.
NOTAPARTITIONTABLE	Indicates the partition filter expression of a non-partitioned table.

- Select a recommended partition filter expression

  This section uses the dt partition as an example to describe how to select a recommended partition filter expression. We recommend that you specify a regular expression as the partition filter expression for a dynamic partitioned table.

  i. In the **Add Partition** dialog box, click the Partition Expression field. A drop-down list appears to show you the partition filter expressions recommended by Data Quality.

  - Select a recommended partition filter expression if it meets your expectation.

- Specify a custom partition filter expression if no recommended partition filter expressions meet your expectation.

ii. After you enter a partition expression, click **Verify**. Data Quality uses the current time, that is, the data timestamp, to calculate data and verify the partition filter expression.

iii. Click **OK**.

If you need to delete a partition filter expression, move the pointer over the partition filter expression and click the **Delete** icon to delete the partition filter expression. When you delete a partition filter expression, all rules configured based on the partition filter expression are also deleted.

Link a partition filter expression to a node

To monitor the quality of data involved in a node, you need to link a partition filter expression to the node.

- The Manage Linked Nodes dialog box lists all committed nodes. Data Quality allows you to link a partition filter expression to a node in another workspace.
- Before you link a partition filter expression to a node in another workspace, make sure that you are an administrator, a developer, or an administration expert in the two workspaces.

You can link a partition filter expression to one or more nodes. After nodes are linked, Data Quality can automatically monitor linked nodes.

> **Note** Data Quality allows you to flexibly link a partition filter expression to a node. You can select a node that is not related to your table.

1. On the rule configuration page of a table, click **Manage Linked Nodes**.
2. In **Manage Linked Nodes** dialog box, enter the name of the node that you want to link to the partition filter expression.
3. Click **Create**.

Create a rule

The Monitoring Rules page is the most important part of Data Quality, where you can create rules for your tables.

Data Quality allows you to create template rules and custom rules as needed. If you need to create a template rule or a custom rule, you can click **Add Monitoring Rule** or **Quick Create**. For more information, see Rules.

After rules are configured, you can click **Batch Create** to save all the configured rules for the current partition filter expression.

Creation method	Parameter	Description
	Rule Name	The name of the rule.

Creation method	Parameter	Description
Add Monitoring Rule	Rule Type	The type of the rule. Valid values: • **Rule Type**: If a node reaches the error threshold, Data Quality reports an error alert and determines that the node fails. If a node reaches the warning threshold, Data Quality reports a warning alert and determines that the node is successful. • **Soft**: If a node reaches the error threshold, Data Quality reports an error alert and determines that the node is successful. If a node reaches the warning threshold, Data Quality does not report a warning alert and determines that the node is successful.
	Auto-Generated Threshold	You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher.
	Rule Source	The source of the rule. Valid values: **Built-in Template** and **Rule Templates**.
	Field	The fields to be monitored. You can select **All Fields in Table** or a specific field. If you select a field, you can apply the rule to the specified field in the table. **Note** In this example, select All Fields in Table and set other parameters for the table-specific rule.
	Template	• If you set **Rule Source** to **Built-in Template**, the built-in table-specific rules appear. • If you set **Rule Source** to **Rule Templates**, you must set parameters, such as **Sampling Method** and **Set Flag**.
	Comparison Method	The comparison method of the rule. Valid values: **Absolute Value**, **Raise**, and **Drop**.
	Thresholds	The warning threshold and error threshold of the fluctuation. You can adjust the slider to specify thresholds or directly enter thresholds.
	Description	The description of the rule.
	Rule Name	The name of the rule.

Creation method	Parameter	Description
Quick Create	Field	The fields to be monitored. You can select All Fields in Table or a specific field. If you select a field, you can apply the rule to the specified field in the table.
	Trigger	• The trigger condition of the rule. If you select All Fields in Table for the Field parameter, you can set this parameter to The number of columns is greater than 0 or Table row number dynamic threshold. Notice You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher. • If you select a field for the Field parameter, you can select The field value already exists, Null Field, or Unique value dynamic threshold. Notice You can use the dynamic threshold feature only in DataWorks Enterprise Edition or higher.

Test rules

After rules are configured for a partition filter expression, you can test all these rules and view the test results.

Note You can manually run these rules to test their configuration and notification methods. We recommend that you test rules as required.

1. On the rule configuration page of a table, click **Test**.
2. In the **Test** dialog box, set the **Data Timestamp** parameter.

Parameter	Description
Partition	The partition filter expression for which rules are run. The actual partition key varies with the data timestamp. For a non-partitioned table, use NOPARTITIONTABLE as the partition filter expression.
Data Timestamp	The data timestamp for testing rules. The default value is the current time.

3. Click **Test**.
4. In the Test dialog box, click **The test is complete. Click to view the results** to view the test results on the **Node Query** page.

Manage subscriptions

By default, Data Quality sends notifications to the user who created a partition filter expression. You can add other users so that Data Quality sends notifications to them.

1. On the rule configuration page of a table, click **Manage Subscriptions**.
2. In the **Manage Subscriptions** dialog box, specify the notification method and notification receiver.Data Quality supports the following four methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.

   > **Note** Add a DingTalk chatbot and obtain a webhook URL. Then, copy the webhook URL to the Manage Subscriptions dialog box.

3. Click **Save**.

View operational logs

On the rule configuration page of a table, click **View Operation Log**. In the **Operations Logs** right-side pane, you can view the information about each operation, including the user who performed the operation, the time when the operation was performed, and the operation details.

The **Details** column displays the details of each operation performed on the current partition filter expression, including the rule configuration details.

View check results

On the rule configuration page of a table, click **View Check Results** to go to the **Node Query** page. On this page, you can view the check results for all rules under the current partition filter expression.

Clone rules

1. On the rule configuration page of a table, click **Clone Rules**.
2. In **Clone Rules** dialog box, set the **Target Expression** parameter.
3. Select **Clone Subscribers** or **Change Table Names in Custom Rules** as required.
4. Click **Clone**.

2.10.3.2. Configure monitoring rules for DataHub

The Monitoring Rules page is the most important part of Data Quality, where you can configure rules to monitor data in E-MapReduce, MaxCompute, and DataHub. This topic describes how to configure monitoring rules for DataHub.

Context

DataHub monitoring supports the following features:

- Templates for monitoring stream discontinuity and data latency
- Stream processing features, such as custom Flink SQL, dimension table JOIN, multi-stream JOIN, and window functions

Procedure

1. Add a DataHub connection.
   i. Log on to the DataWorks console.
   ii. On the **DataStudio** page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Integration**.
   iii. On the Data Integration page, click **Connection** in the left-side navigation pane. The **Data Source** page appears.
   iv. Click **Add Connection** in the upper-right corner to add a DataHub connection.
2. Select the DataHub connection.
   i. On the current page, click  in the upper-left corner and choose **All Products > Data Quality**.
   ii. On the Data Quality page, click **Monitoring Rules** in the left-side navigation pane.
   iii. Select **Datahub** from the **Engine/Data Source** drop-down list and select the DataHub connection. All the topics in the selected DataHub data store appear.

Parameter	Description
Configure Flink Resources	After you add a connection, click Configure Flink Resources to configure Flink and Log Service resources related to the connection.
Topics	The Topics tab lists all topics in the DataHub data store. You can click the following buttons in the Actions column for a topic: ■ **View Monitoring Rules**: Click it to create rules for the topic. You can create template rules and custom rules as needed. ■ **Manage Subscriptions**: Click it to view and modify subscribers to the current topic, and change the notification method. You can configure the DingTalk chatbot notification method. The changed notification method takes effect for all subscribers to the topic.

Parameter	Description
Dimension Tables	When you create custom rules for a topic, you can create dimension tables and use the JOIN clause to join dimension tables. If the collected data streams lack some fields for a dimension table, you must supplement fields to data streams before data analysis and declare the dimension table in Data Quality. DataHub supports the dimension tables of ApsaraDB for HBase, Lindorm, ApsaraDB for RDS, Tablestore, Taobao Distributed Data Layer (TDDL), and MaxCompute. Flink SQL does not design the DDL syntax for dimension tables. You can use the standard CREATE TABLE statement. However, you must add `period for system_time` to specify the period of a dimension table and declare that the dimension table stores time-varying data. **Note** When you declare a dimension table, you must specify the primary key. When you join a dimension table with another table, the ON condition must contain an equivalence condition that includes the primary key of either table.

iv. Click the **Topics** tab. Find the target topic and click **View Monitoring Rules** in the Actions column.

3. On the rule configuration page of the topic, click **Create rules**.
4. Create a monitoring rule.In Data Quality, you can create template rules and custom rules as needed.
   - Click **Create Template Rule**. Two templates are available: **Data Delay** and **Stream Discontinuity**.

     For example, you can select **Data Delay** for Template Type.

Parameter	Description
Rule Name	The name of the rule. The name can be up to 255 characters in length.
Field Type	The fields to be monitored. By default, the field type is All Fields in Table.

Parameter	Description
Template Type	■ **Data Delay**: monitors the interval between the time when data is generated and the time when data is written to DataHub based on the data timestamp field. If the interval exceeds a specified threshold, an alert is generated. Note ■ Before you configure a stream discontinuity rule, you must activate Realtime Compute in Flink and create a project. ■ The data timestamp field supports two data types: TIMESTAMP and STRING (yyyy -MM -dd H dd HH:mm:ss). ■ **Stream Discontinuity**: monitors the period during which no data is written to DataHub. If the period exceeds a specified threshold, an alert is generated.
Alerts Threshold	The maximum number of alerts generated for data latency. Data Quality reports an alert when the number of alerts generated for data latency exceeds this threshold. This parameter only takes effect when you select Data Delay for Template Type.
Data Timestamp Field	The data timestamp field of the topic for which the rule is created. This field supports two data types: TIMESTAMP and STRING (yyyy-MM-dd HH:mm:ss). This parameter only takes effect when you select Data Delay for Template Type.
Alert Frequency	The interval for reporting an alert. You can set the alert interval to 10 minutes, 30 minutes, 1 hour, or 2 hours.
Warning Threshold	The warning threshold, in seconds. The value must be an integer and less than the error threshold.
Error Threshold	The error threshold, in seconds. The value must be an integer and greater than the warning threshold.

- If template rules do not meet your requirements for monitoring the data quality of DataHub topics, you can click **Create Custom Rule** to create a rule as required.

  Note
  - The field in the SELECT clause must be a column. Ensure that you can compare the field values with the warning and error thresholds.
  - The FROM clause must include the current topic and all its columns.

Parameter	Description

Parameter	Description
Rule Name	The name of the rule. The name must be unique in the topic and can be up to 20 characters in length.
Script	The custom SQL script, which is used to set a rule. The return value of the SELECT clause must be unique. You can refer to the following sample statements: ■ Use a simple SQL statement. `select id as a from zmr_tst02;` ■ Join the topic and a dimension table named test_dim. `select e.id as eid` `from zmr_test02 as e` `join test_dim for system_time as of proctime() as w` `on e.id=w.id` ■ Join the topic and another topic named dp1test_zmr01. `select count(newtab.biz_date) as aa` `from (select o.*` `from zmr_test02 as o` `join dp1test_zmr01 as p` `on o.id=p.id)newtab` `group by id.biz_date,biz_date_str,total_price,'timestamp'`
Warning Threshold	The warning threshold, in minutes. The value must be an integer and less than the error threshold.
Error Threshold	The error threshold, in minutes. The value must be an integer and greater than the warning threshold.
Minimum Alert Interval	The minimum interval for reporting an alert, in minutes.
Description	The description of the rule.

5. Click **Batch Create** to add the created rules to the topic.
   - **View Log**: Click it to view the operational logs of rules.

- **Manage Subscriptions**: Click it to view and modify subscribers to the current rule, and change the notification method. The changed notification method takes effect for all subscribers to the rule.

  Data Quality supports the following four methods: **Email**, **Email and SMS**, **DingTalk Chatbot**, and **DingTalk Chatbot @ALL**.

  > **Note** Add a DingTalk chatbot and obtain a webhook URL. Then, copy the webhook URL to the Manage Subscriptions dialog box.

2.11. Data Map

2.11.1. Overview

Developed based on Data Management, Data Map uses roles to control the permissions for using different features, such as the permissions for creating and previewing data. Data Map helps you build a better enterprise-level knowledge base.

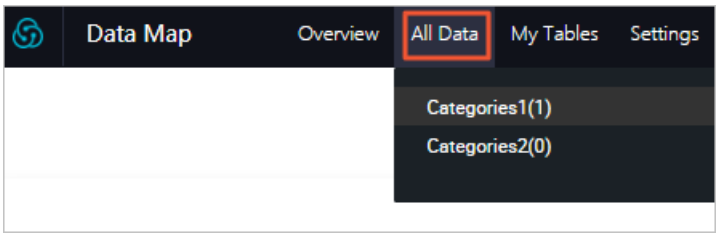
On the homepage of Data Map, you can enter keywords to search for tables by name. You can also click a table in any of the following sections to view the table data: **Recently Viewed Tables**, **Recently Read Tables**, **Most Viewed Tables**, and **Most Read Tables**.

- If you prefer a powerful search engine, go to the homepage to search for data.

  > **Note** The homepage appears by default when you access **Data Map**. To return to the homepage from other pages, click **Data Map** in the upper-left corner.

- If you need to find tables by project or cluster, click **All Data** in the top navigation bar. On the All Data page, you can view tables on the **MaxCompute** and **E-MapReduce** tabs. On the MaxCompute tab, you can perform the following operations on a table: **Add to Favorites**, **Apply for Permission**, **View Lineage**, and **View DDL Statement**.

  If you have added tables to categories, move the pointer over **All Data** in the top navigation bar and select a category. Tables in the category appear. For more information, see Manage categories and configure workspace permissions.

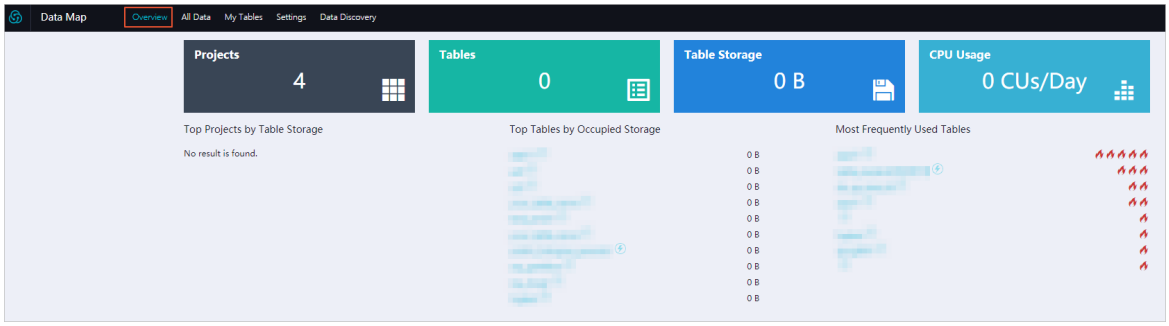
- If you need to view the overall data of the current tenant, click **Overview** in the top navigation bar. For more information, see View overall data.
- If you need to modify tables owned by yourself, click **My Tables** in the top navigation bar. For more information, see View and manage data and data permissions.
- If you are a category administrator or workspace administrator and need to modify the workspace configuration or global categories, click **Settings**. For more information, see Manage categories and configure workspace permissions.

2.11.2. View overall data

This topic describes how to view the overall data of a tenant on the Overview page.

Procedure

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Map(Data Management)**. The homepage of Data Map appears.
3. Click **Overview** in the top navigation bar. The **Overview** page displays offline statistics of the tenant.



> **Note** The Overview page displays statistics of the previous day.

Section	Description
Projects	The total number of projects of the tenant.
Tables	The total number of tables of the tenant.
Table Storage	The total storage occupied by all tables of the tenant.
CPU Usage	The number of compute units (CUs) consumed by the tenant in one day. One CU is equivalent to the computing resources consumed by one fully loaded CPU core in one day.
Top Projects by Table Storage	The top projects that occupy the most storage space under the tenant.

Section	Description
Top Tables by Occupied Storage	The top tables that occupy the most storage space under the tenant. You can click a table name to go to the details page of the table. Note The logical storage space occupied by projects and tables is collected in a T+1 manner. The numbers next to the project and table names indicate the sizes of the occupied logical storage space. The project storage volume includes the table storage volume, storage volumes of resources, data in the recycle bin, and other system files. Therefore, the project storage volume is larger than the table storage volume. The table storage volume is charged based on the logical storage rather than the physical storage.
Most Frequently Used Tables	The most frequently referenced tables of the tenant. You can click a table name to go to the details page of the table.

2.11.3. View and manage data and data permissions

You can view and manage data on the Owned by Me, Managed by Me as Workspace Administrator, Managed by Tenant Account, and My Favorites pages. This topic describes how to view and manage data and data permissions.

Context

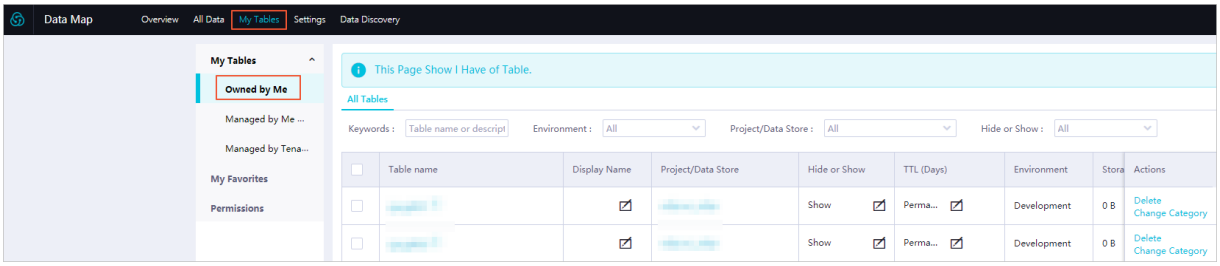
Data Map updates data one day after the data is generated. If you need to query real-time data, we recommend that you use SQL statements.

Go to the My Tables page

1. Log on to the DataWorks console.
2. Click  in the upper-left corner and choose **All Products > Data Map(Data Management)**. The homepage of Data Map appears.
3. In the top navigation bar, click **My Tables**. By default, the **My Tables > Owned by Me** page appears.

View and manage data on the Owned by Me page

On the **Owned by Me** page, you can search for data by keyword, environment, project or data store, and visibility. You can also view the details about a table and perform relevant operations on the table.

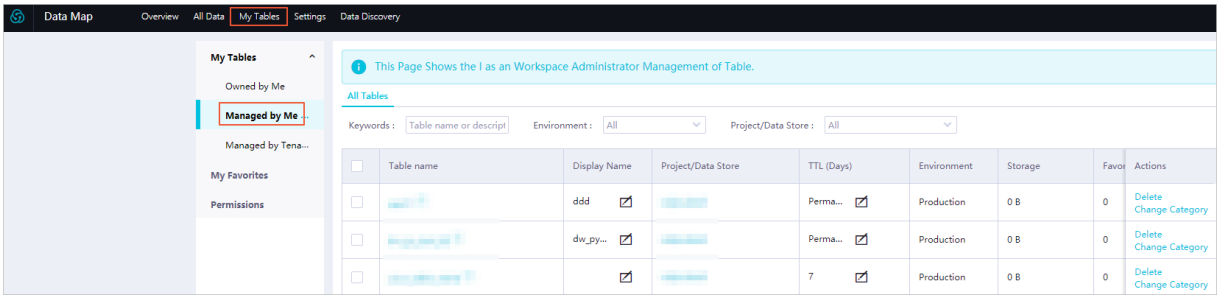


GUI element	Description
Table name	The name of the table. You can click a table name to go to the table details page.
Display Name	The display name of the table. You can click in the **Display Name** column of a table to change the display name of the table.
Project/Data Store	The project or data store where the table resides.
Hide or Show	Indicates whether the table is hidden. You can click in the **Hide or Show** column of a table to hide or show the table. **Note** If you hide a table, the **Apply for Permissions** button does not appear on the details page of the table.
TTL (Days)	The TTL of the table, which is the same as that you set when you create the table.
Environment	The environment to which the table belongs. Valid values: **Development** and **Production**.
Storage	The amount of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days.
Created At	The time when the table was created.
Actions	The operations that you can perform on the table. You can click **Delete** or **Change Category** in the Actions column of a table to delete the table or change the category of the table.

GUI element	Description
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and then click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the table owners, delete the tables, or change the categories of the tables.

View and manage data on the Managed by Me as Workspace Administrator page

In the left-side navigation pane, click **Managed by Me as Workspace Administrator**. On the page that appears, you can search for data by keyword, project or data store, and environment. You can also view the details about a table and perform relevant operations on the table.

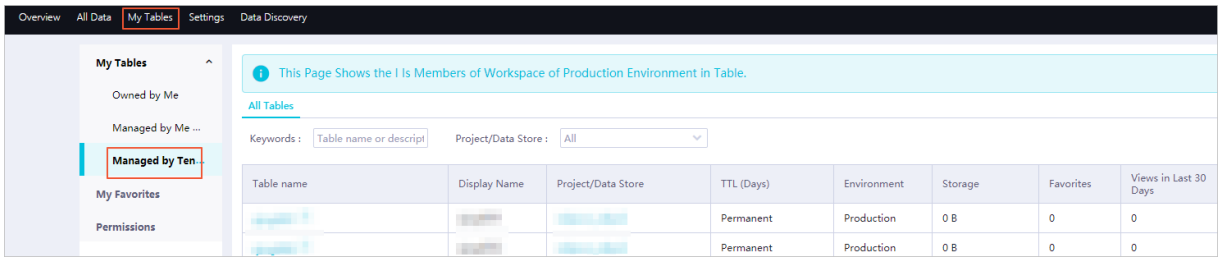


GUI element	Description
Table name	The name of the table. You can click a table name to go to the table details page.
Display Name	The display name of the table. You can click in the **Display Name** column of a table to change the display name of the table.
Project/Data Store	The project or data store where the table resides.
TTL (Days)	The TTL of the table, which is the same as that you set when you create the table.
Environment	The environment to which the table belongs. Valid values: **Development** and **Production**.
Storage	The amount of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days.
Created At	The time when the table was created.
Actions	The operations that you can perform on the table. You can click **Delete** or **Change Category** in the Actions column of a table to delete the table or change the category of the table.

GUI element	Description
Edit, Change Owner, Delete, and Change Category	The operations that you can perform on multiple tables at a time. You can select tables and then click **Edit**, **Change Owner**, **Delete**, or **Change Category** to modify the tables, change the table owners, delete the tables, or change the categories of the tables.

View data on the Managed by Tenant Account page

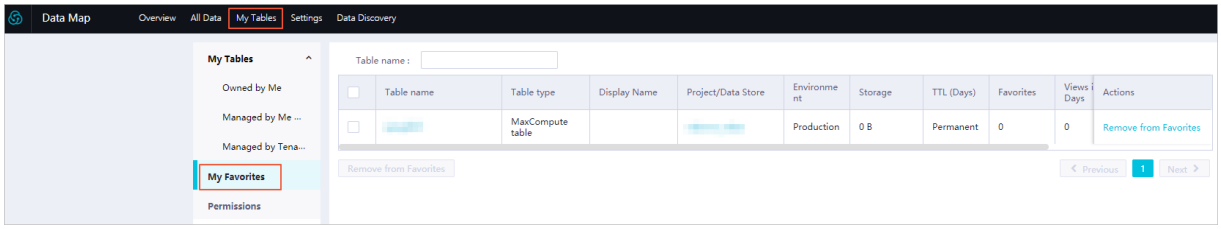
In the left-side navigation pane, click **Managed by Tenant Account**. On the page that appears, you can search for data by keyword and project or data store. You can also view the details about a table and perform relevant operations on the table.



GUI element	Description
Table name	The name of the table. You can click a table name to go to the table details page.
Display Name	The display name of the table. You can click in the **Display Name** column of a table to change the display name of the table.
Project/Data Store	The project or data store where the table resides. Tables have different suffixes when they are deployed in different environments. For example, **_dev** indicates the development environment.
TTL (Days)	The TTL of the table, which is the same as that you set when you create the table.
Environment	The environment to which the table belongs. Valid values: **Development** and **Production**.
Storage	The amount of data that is stored in the table.
Favorites	The number of times that users add the table to favorites.
Views in Last 30 Days	The number of times that users view the table in the last 30 days.
Created At	The time when the table was created.

View data on the My Favorites page

In the left-side navigation pane, click **My Favorites**. On the page that appears, you can view the tables that you have added to favorites.



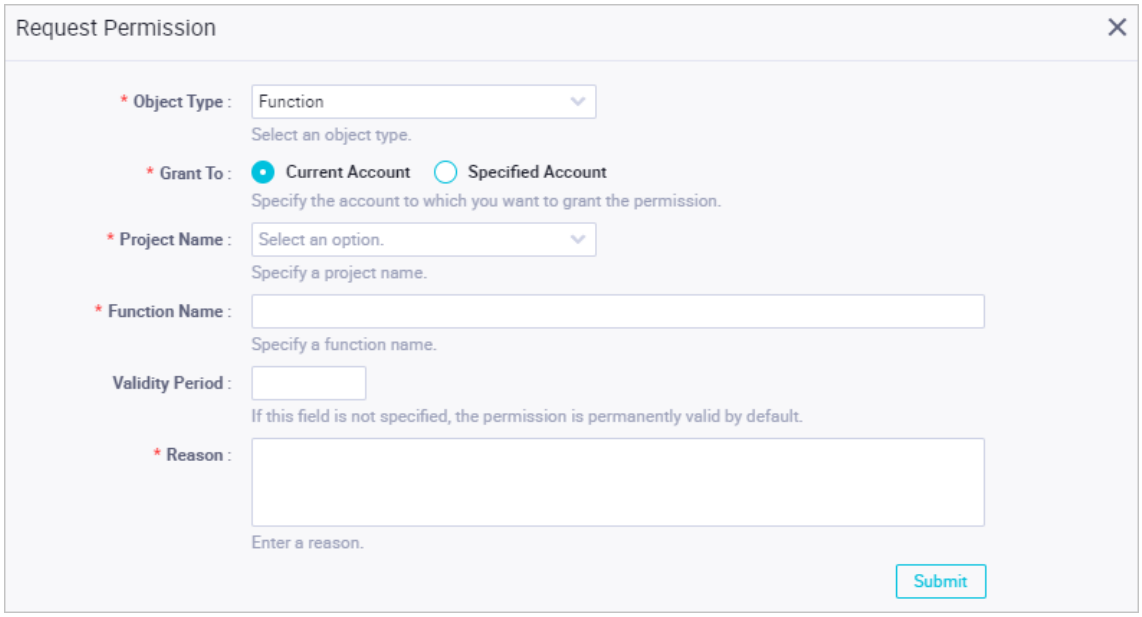
You can click **Remove from Favorites** in the Actions column of a table to remove the table from your favorites.

View and manage data permissions

In the left-side navigation pane, click **Permissions**. On the Permissions page, you can view and manage data permissions.

You can click **Apply function and resource permissions** in the upper-right corner of the **Permissions** page to request permissions. You can also view permission requests on the **To Be Approved**, **Submitted by Me**, and **Handled by Me** tabs.

- **Apply function and resource permissions**
  i. On the Permissions page, click **Apply function and resource permissions** in the upper-right corner.
  ii. In the **Request Permission** dialog box, set the parameters as required. The following table describes the parameters for requesting permissions.



Parameter	Description
Object Type	The type of object on which you want to request permissions. Valid values: **Function** and **Resources**.

Parameter	Description
Grant To	The account to which the permissions will be granted. Valid values: **Current Account** and **Specified Account**. ■ If you select **Current Account**, the permissions will be granted to you after the request is approved. ■ If you select **Specified Account**, you must also specify the **Account** parameter. The permissions will be granted to the specified account after the request is approved.
Project Name	The name of the MaxCompute project that contains the target function or resource.
Function Name or Resource Name	The full name of the function or resource in the project. If the resource is a file, enter the full name of the file, including the file name extension. Example: my_mr.jar.
Validity Period	The validity period of the permissions, in days. If this parameter is not specified, the permissions are permanently valid. After the validity period expires, the system automatically revokes the permissions.
Reason	The reason why you request the permissions.

- **To Be Approved**

  If you are the workspace administrator, you can go to the **To Be Approved** tab. On the tab, you can view and approve the requests for permissions on all objects such as tables, resources, and functions in the workspace.

- **Submitted by Me**

  Click the **Submitted by Me** tab on the **Permissions** page.

  You can view the permission requests that you have submitted on this tab.

- **Handled by Me**

  If you are the workspace administrator, you can go to the **Handled by Me** tab on the **Permissions** page.

  On the **Handled by Me** tab, you can view the permission requests that you have handled for all objects such as tables, resources, and functions in the workspace.

2.11.4. Manage categories and configure workspace permissions

This topic describes how to manage categories and configure workspace permissions on the Settings page.

Go to the Settings page

1. Log on to the DataWorks console.
2. Click  in the upper-left corner and choose **All Products > Data Map(Data Management)**. The

homepage of Data Map appears.

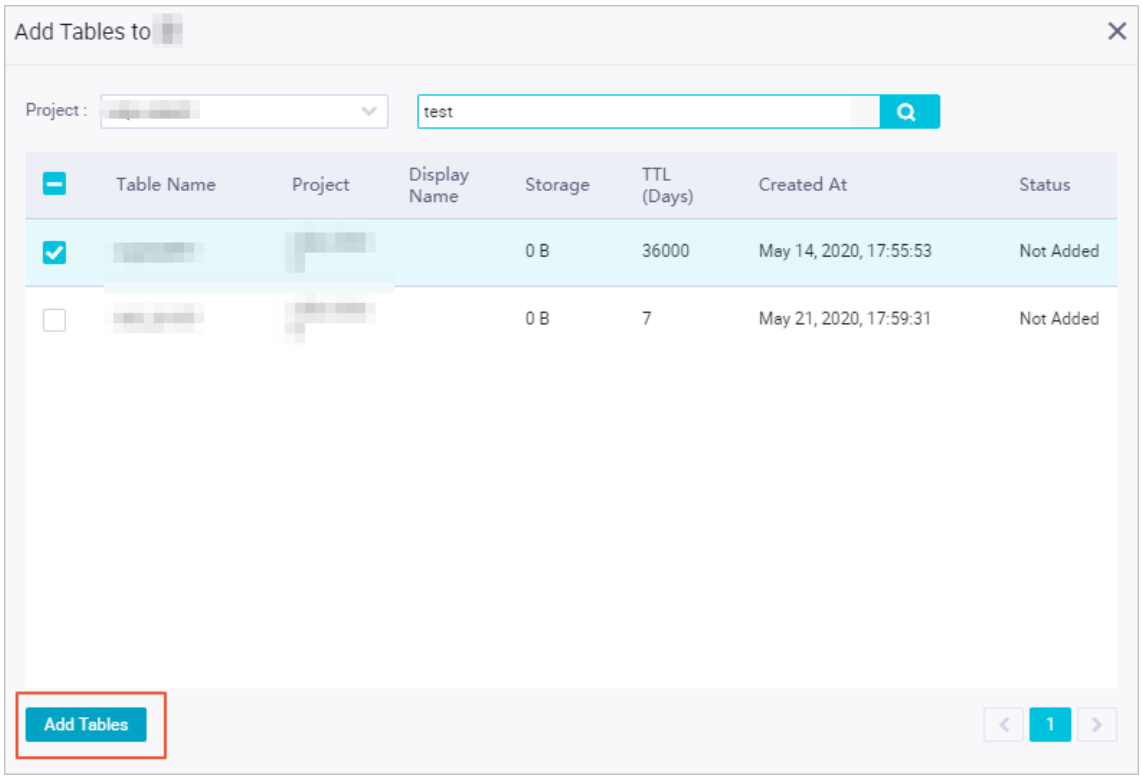
3. In the top navigation bar, click **Settings**. The **Manage Categories** page appears by default.

Manage categories

Only tenant administrators have permissions to create categories and add tables to the categories on the **Manage Categories** page. For more information about how to assign the tenant administrator role to a user, see Member management.

To manage categories as a tenant administrator, perform the following steps:

1. On the **Manage Categories** page, move the pointer over **Categories** and click + to create a level-1 category.
2. Move the pointer over the name of the level-1 category and click + to create a level-2 category.Use the same method to create more categories. DataWorks allows you to create categories at a maximum of four levels. You can change category names and delete categories.
   - To change the name of a category, move the pointer over the category name, click , enter a new name in the field that appears, and then press Enter.
   - To delete a category, move the pointer over the category name and click . In the **Delete Category** message, click **OK**.
3. Add tables to and remove tables from a category.
   - To add tables to a category, select the category and click **Add Tables** in the upper-right corner. In the dialog box that appears, select the required tables and click **Add Tables**.



   - To remove a table from a category, select the category, find the target table, and then

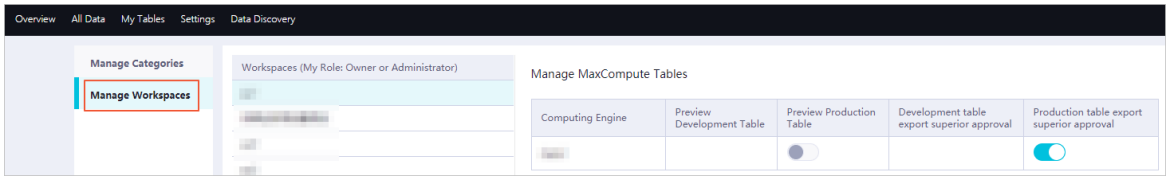
click **Remove from Category** in the Actions column. In the message that appears, click **OK**.

> **Note** To remove multiple tables from a category, select the category, select the target tables, and then click **Remove from Category** in the lower part of the table list.

Configure workspace permissions

To configure workspace permissions on the Settings page, perform the following steps:

1. In the left-side navigation pane, click **Manage Workspaces**.
2. Click the name of the target workspace.
3. In the **Manage MaxCompute Tables** section, configure the permissions on the workspace.You can turn on or off the **Preview Development Table** and **Preview Production Table** switches as required.



> **Note** If the workspace is in basic mode, you can turn on or off only the **Preview Production Table** switch.

2.11.5. View table details

2.11.5.1. View the details of a table

This topic describes how to go to the details page of a table and view the details about the table, such as the basic information, output information, and lineage information.

Go to the details page of a table

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Map(Data Management)**. The homepage of Data Map appears.
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab as required.
5. On the tab that appears, click the name of the table you want to view.On the details page that appears, you can view the basic information, business information, permission information, technical information, detailed information (including the fields, partitions, and change history), output information, lineage information, reference records, and usage notes of the table. You can also preview data in the table.

View basic information

In the **Basic Information** section, you can view the numbers of reads, favorites, and views. You can also check the output nodes, MaxCompute project name, region to which the current workspace belongs, region to which the engine belongs, owner, creation time, time-to-live (TTL), storage capacity, description, and tags of the table, and whether the table is partitioned.

You can perform the following operations in this section:

- View the code of the output node of the table: Click **View Code** next to **Output Node**. On the **Operation Center** page, view the node code.
- View the details about the MaxCompute project: Click the MaxCompute project name. On the page that appears, view the details about the MaxCompute project to which the table belongs.
- Edit the description of the table: Click next to **Description**, enter a description in the field that appears, and then click .
- Add a tag to or remove a tag from the table: Click next to **Label**, enter a tag name in the field that appears, and then press **Enter**.

  To remove a tag from the table, move the pointer over the tag and click the **Remove** icon.

View business information

In the **Business Information** section, you can view the DataWorks workspace name, environment type, and category of the table.

In this section, you can click the workspace name next to **DataWorks Workspace** to view the details about the workspace to which the table belongs.

View permission information

In the **Permission Information** section, you can view your permissions on the table.

To modify your permissions on the table, perform the following steps:

1. Click **More** in the upper-right corner of the **Permission Information** section.
2. In the **Apply for Data Permissions** dialog box, set the **Grant To**, **Validity Period**, and **Reason** parameters as required.

   > **Note** If you do not set the **Validity Period** parameter, the permissions that you request will be permanently valid after your request is approved.

3. Click **Commit**.

View technical information

In the **Technical Information** section, you can view the technical type, last time when the DDL statement was modified, last time when the data was modified, last time when the data was viewed, and compute engine information.

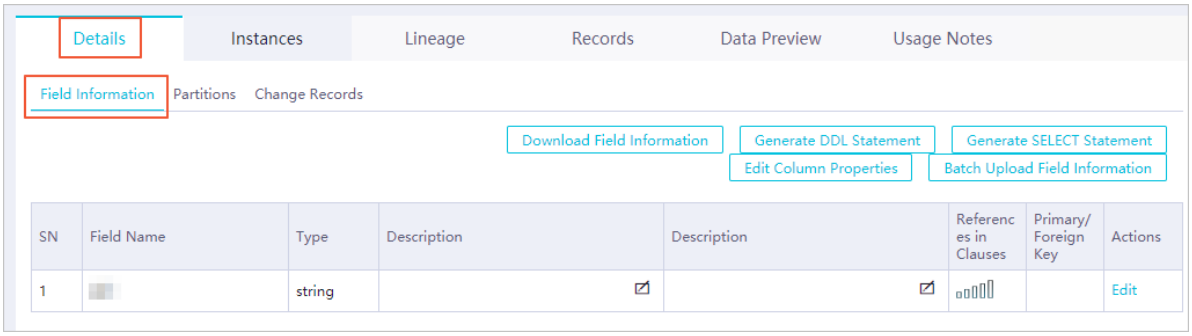
In this section, you can click **Click to View** next to **Compute Engine Information**. In the **Compute Engine Information** dialog box, you can view or copy the information about the compute engine.

View detailed information

The **Details** tab contains the following tabs: **Field Information**, **Partitions**, and **Change Records**.

- **Field Information tab**

  On the **Field Information** tab, you can view the name, data type, description, business description, and popularity of fields. You can also check whether a field is a primary key or foreign key.



Button	Description
Edit Column Properties	Click this button, modify the field description and business description, and then click **Save** or **Cancel** as needed.
Batch Upload Field Information	Click this button and drag the local file to be uploaded to the **Batch Upload Field Information** dialog box.
Download Field Information	Click this button to download the field information of the current table.
Generate DDL Statement	Click this button. In the **Generate DDL Statement** dialog box, view or copy the DDL statement used to create the current table.
Generate SELECT Statement	Click this button. In the **Generate SELECT Statement** dialog box, view or copy the `SELECT` statement for querying data in the current table.

- **Partitions tab**

  On the **Partitions** tab, you can view the name, number of records, storage capacity, creation time, and last update time of each partition in the current table.

- **Change Records tab**

  On the **Change Records** tab, you can view the description, type, granularity, time, and operator of changes performed on the current table.

  On this tab, you can also select a change type from the drop-down list in the upper-left corner to filter the table changes.

  Change types include **Create Table**, **Modify Table**, **Delete Table**, **Add Partition**, **Delete Partition**, **Change Owner**, and **Modify Lifecycle**.

View output information

If the table data periodically changes with the corresponding node, you can view the change status and data that is continuously updated on the **Instances** tab.

On this tab, you can also click **Code** or **Logs** in the **Actions** column of a node to view the code or logs of the node.

View lineage information

On the Lineage tab, you can view the source and destination of data and manage the lineage information with ease.

The **Lineage** tab contains the following tabs: **Table Lineage**, **Field Lineage**, and **Impact Analysis**.

- **Table Lineage** tab: On this tab, you can search for the ancestor and descendant tables of the current table based on the globally unique identifier (GUID).
- **Field Lineage** tab: On this tab, you can select a field from the **Field Name** drop-down list to view the lineage information of the field.
- **Impact Analysis** tab: On this tab, you can query the node that generates a lineage and the full link of the lineage based on information such as the field, node type, table name, project name, and table owner.

You can perform the following operations on the query result:

- Click **Download** to download the query result to your local computer.
- Click **Email**. In the **Notify by Email** dialog box, set the required parameters and click **OK**. In this way, you can configure DataWorks to send email notifications to owners of the descendant tables of the current table.

View reference records

The **Records** tab contains the following tabs: **Foreign Key References** and **Access Statistics**.

- **Foreign Key References** tab: On this tab, you can check the number of users who reference the current table.
- **Access Statistics** tab: On this tab, you can view the reference records in a line chart.

Preview data

On the **Data Preview** tab, you can preview the data of the current table.

> **Notice** Only authorized users can preview tables in the production environment. If you do not have the required permissions, click **Apply Now**.

View usage notes

On the **Instructions** tab, you can edit usage notes, check the historical versions of the usage notes, and view the business description of data.

2.11.5.2. Request table permissions

This topic describes how to request table permissions on the Data Map or Security Center page.

Go to the details page of a table

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Map(Data Management)**. The homepage of Data Map appears.

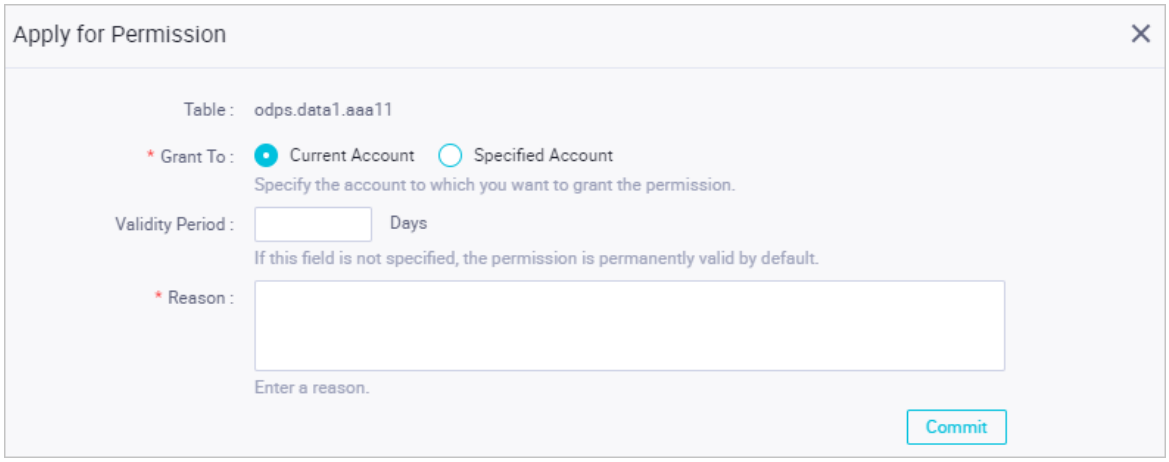
3. In the top navigation bar, click **All Data**.
4. On the All Data page, click a tab as required.
5. On the tab that appears, click the name of the table on which you want to request permissions.

Request table permissions

1. On the table details page, click **Apply for Permission**.

   > **Note** If a table is hidden, the **Apply for Permissions** button does not appear on the details page of the table.

2. In the **Apply for Permission** dialog box, set the parameters as required. The following table describes the parameters for requesting table permissions.



Parameter	Description
Table	The table on which you want to request permissions. You cannot change the default value.
Grant To	The type of the account to which the permissions will be granted. Valid values: **Current Account** and **Specified Account**.
Username	The username of the account to which the permissions will be granted. > **Note** This parameter is available only when you set the **Grant To** parameter to **Specified Account**.
Validity Period	The validity period of the permissions. If you do not set this parameter, the permissions are permanently valid.
Reason	The reason why you request the permissions. Enter a reason for faster approval.

3. Click **Commit**.

View the request status

1. Click in the upper-left corner and choose **All Products > Data Map(Data Management)**. The homepage of Data Map appears.
2. In the top navigation bar, click **My Tables**.
3. On the My Tables page, click **Permissions** in the left-side navigation pane.
4. On the page that appears, click the **Submitted by Me** tab.
5. Find the target request record and click **View** in the Actions column. The request details appear.

2.11.5.3. Add a table to favorites

This topic describes how to add a table to or remove it from favorites, and view the tables added to favorites.

Procedure

1. Log on to the DataWorks console.
2. Go to the details page of a table and click **Add to Favorites** in the upper-left corner.
3. In the top navigation bar, click **My Tables**.
4. On the **My Tables** page, click **My Favorites** in the left-side navigation pane. On the page that appears, you can view all the tables that you have added to favorites and remove tables from favorites. To remove a table from favorites, find the table in the table list and click **Remove from Favorites** in the Actions column.

2.12. Data Asset Management

2.12.1. Go to the Data Asset Management page

Data Asset Management provides you with an overview of your data assets. Data Asset Management requires that data be synchronized by using Data Integration and processed by using DataStudio before you manage your tables and APIs stored in your business system and DataWorks.

Context

Data Asset Management controls the permissions of users independently. You must grant the permissions on the Project Management page because Data Asset Management is a tenant-level feature.

Data Asset Management allows you to view the metadata collected in Data Map. You can also perform basic management operations on the metadata. For example, you can change the business classes and add business descriptions for metadata tables.

Procedure

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Project Management**. The Project Management page appears.

3. In the left-side navigation pane, click **Member Management**. On the Member Management page, you can assign the following roles to members: **Data Asset Management-Asset Manager**, **Data Asset Management-Class Manager**, and **Data Asset Management-Home Visitor**.
4. Click in the upper-left corner and choose **All Products > Data Asset Management** to manage data assets in Data Asset Management. The following table describes the permissions of each role.

Role	Permission
Data Asset Management-Asset Manager	In the top navigation bar of the **Data Asset Management** page, click the **Assets** tab. On the Assets tab, you can manage data assets, such as adding business units and classes. You can also add data assets to a class.
Data Asset Management-Class Manager	In the top navigation bar of the **Data Asset Management** page, click the **Classes** tab. On the Classes tab, you can view the data assets of each class.
Data Asset Management-Home Visitor	In the top navigation bar of the **Data Asset Management** page, click the **Home** tab. On the Home tab, you can view the statistics of data assets of different classes and business units.

2.12.2. Asset manager

Asset managers can view the information about data assets.

Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. On the **Home** page that appears, enter keywords in the search box and click **Search**.
4. On the search results page that appears, click the **Tables**, **File**, or **API** tab to view details and apply for permissions.

   You can click the **Classes** tab in the top navigation bar to filter data assets by class.

2.12.3. Asset user

Asset users can access Data Asset Management to perform operations such as searching for assets, applying for permissions, and using assets.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. On the **Home** page that appears, enter keywords in the search box and click **Search**.
4. On the search results page that appears, click the **Tables**, **File**, or **API** tab to view details and apply for permissions.

You can click the **Classes** tab in the top navigation bar to filter data assets by class.

2.12.4. Asset administrator

Asset administrators can manage assets and authorizations in Data Asset Management.

> **Note** You can submit a ticket to apply for the asset administrator role.
>
> An administrator can grant the administrator role to common users. An administrator can perform any operations in Data Asset Management, and no approval is required.

Go to the Assets tab

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
3. In the top navigation bar, click the **Assets** tab.

   Under the **Assets** tab, you can click **Data Management**, **Classes**, or **Business Management** in the left-side navigation pane to manage your data assets accordingly.

Manage data

In the left-side navigation pane, you can click **Data Management** and then **Tables**, **Files**, or **APIs** to manage tables, files, or APIs.

- **Manage tables**

  In the left-side navigation pane, choose **Data Management > Tables** to go to the **Tables** page.

  On the Tables page, you can view, edit, publish, or delete a table.

  - Click the name of a table to view table details.
  - Click **Edit** in the Actions column of a table. In the Edit dialog box that appears, you can edit the configuration of the table.
  - Move the pointer over **Publish** in the Actions column of a table. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You can search for a published table in Data Asset Management.

  - Move the pointer over **Delete** in the Actions column of a table. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

    > **Note** You cannot search for a deleted table in Data Asset Management.

- **Manage files**

  In the left-side navigation pane, choose **Data Management > Files** to go to the **Files** page.

On the **File Management** page that appears, you can upload a file. Then, you can view, edit, download, or delete the file.

- In the upper-right corner, click **Upload File**. In the **Upload File** dialog box that appears, click **Add File**. In the Add dialog box that appears, select the file to be uploaded and click **Open**. Alternatively, you can drag and drop a file to the **Upload File** dialog box. Then, click **Next**.

  > Note
  > - To upload a file, make sure that the size of the file does not exceed 50 MB.
  > - You can only upload a file with one of the following file name extensions:
  >
  >   3DX, 7Z, A3D, ATX, AVI, BMP, SV, DBF, DOC, DOCX, DWG, EPS, ESP, FREELIST, GDB, GDBINDEXES, GDBTABLE, GDBTABLX, GIF, GZ, HTM, HTML, IVE, JPEG, JPG, LOCK, LSP, LST, MP3, MP4, MPJ, OSG, OSGB, PDF, PNG, PPT, PPTX, PRJ, PSD, RAR, S3C, SBN, SBX, SCP, SHP, SPX, TFW, TIF, TIFF, TTF, TXT, WAV, WL, WP, WT, XLS, XLSX, ZIP, XML, SHX, and SKP

- Click the name of a file to view file details.
- Click **Edit** in the Actions column of a file. In the Edit dialog box that appears, you can edit the configuration of the file.
- Move the pointer over **Publish** in the Actions column of a file. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > Note You can search for a published file in Data Asset Management.

- Move the pointer over **Delete** in the Actions column of a file. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > Note You cannot search for a deleted file in Data Asset Management.

- Click **Download** in the Actions column of a file to download the file.

  > Note Before downloading a file, apply for the download permission.

- **Manage APIs**

  In the left-side navigation pane, choose **Data Management** > **APIs** to go to the **APIs** page.

  On the **APIs** page, you can edit, publish, or delete an API.

  - Click **Edit** in the Actions column of an API. In the **Edit** dialog box that appears, you can edit the configuration of the API. Then, click **Submit**.

- Move the pointer over **Publish** in the Actions column of an API. In the dialog box that appears, click **Publish**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > **Note** You can search for a published API in Data Asset Management.

- Move the pointer over **Delete** in the Actions column of an API. In the dialog box that appears, click **Delete**. After the request is submitted, click the **Permissions** tab in the top-navigation bar. In the left-side navigation pane, click **Submitted by Me** to view the request you submitted.

  > **Note** You cannot search for a deleted API in Data Asset Management.

Manage classes

1. In the left-side navigation pane, click **Classes** to go to the **Classes** page.

   On the Classes page, you can import or export a class.

2. Click the icon. In the **Add Class** dialog box that appears, set relevant parameters and click **OK** to add a level-1 class

Parameter	Description
Name	The name of the class, which can be up to 128 characters in length.
Code	The code of the class. This parameter cannot be left empty.
Description	The description of the class.
Confidential	Specifies whether the class is confidential. Valid values: **Yes** and **No**.
Share	Specifies whether to share the class. Valid values: **Yes**, **Conditional**, and **No**.

   To create a subclass under a class, click the icon next to the class.

3. Click a class. On the page that appears, click the **Tables** tab.
4. Click **Add Table**. In the **Add Table** dialog box that appears, select the tables to be added to the class and click **OK**.

   You can add files and APIs to a class in the same way.

   To change the class of a table, click **Modify Class** in the Actions column. In the **Change Class** dialog box that appears, change the class as needed.

Manage business

In the left-side navigation pane, you can click **Business Management** and then **Business Units**, **Business Systems**, or **Connections** to manage business units, business systems, or connections.

> **Note** Connections belong to a business system, and business systems belong to a business unit.
>
> - A business system with connections cannot be deleted.
> - A business unit with business systems cannot be deleted.

- **Manage business units**

  In the left-side navigation pane, choose **Business Management** > **Business Units** to go to the **Business Units** page.

  Click the icon. In the **Add Business Unit** dialog box that appears, set relevant parameters and click **OK** to add a business unit.

Parameter	Description
Name	The name of the business unit. This parameter cannot be left empty.
Code	The code of the business unit. By default, the code cannot be modified.
Description	The description of the business unit.
Confidential	Specifies whether the business unit is confidential. Valid values: **Yes** and **No**.
Share	Specifies whether to share the business unit. Valid values: **Yes**, **Conditional**, and **No**.
Business system included	Select the business systems to be added to the business unit and click the > icon.

  To create a sub-business unit under a business unit, click the icon next to the business unit.

- **Manage business systems**

  In the left-side navigation pane, choose **Business Management** > **Business Systems** to go to the **Business Systems** page.

  On the Business Systems page, you can add a business system. Then, you can view, edit, or delete the business system.

  - Click **Add Business System**. In the **Basic information** dialog box that appears, set relevant parameters and click **Submit**.
  - Click **View** in the Actions column of a business system to view its details.
  - Click **Edit** in the Actions column of a business system. In the **Business System Properties** dialog box that appears, you can edit the configuration of the business system.
  - Click **Delete** in the Actions column of a business system. In the **Delete business system** dialog box that appears, click **OK** to delete the business system.

- **Manage connections**

  In the left-side navigation pane, choose **Business Management** > **Connections** to go to the **Connections** page.

  On the **Connections** page, you can view the information of connections. The connection information includes the connection name, the number of tables, the owner, the business system to which the connection belongs, the data type, and the update time. You can also edit the configuration of a connection.

2.12.5. Manage authorizations

Under the Permissions tab, you can view permissions in different states on the Submitted by Me, To Be Handled, Handled by Me, and My Permissions pages respectively.

Go to the Permissions tab

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products** > **Data Asset Management**.
3. In the top navigation bar, click the **Permissions** tab.

   Under the **Permissions** tab, you can view permissions in different states on the **Submitted by Me**, **To Be Handled**, **Handled by Me**, and **My Permissions** pages respectively.

Submitted by Me

In the left-side navigation pane, click **Submitted by Me**.

On the **Submitted by Me** page, you can view request details or cancel requests submitted by you. To resubmit a request that was not approved, find the target request and click **Reapply**.

To Be Handled

In the left-side navigation pane, click **To Be Handled**. On the To Be Handled page, you can view request details and approve or reject requests.

Handled by Me

In the left-side navigation pane, click **Handled by Me** to view requests handled by you.

My Permissions

In the left-side navigation pane, click **My Permissions**. On the My Permissions page, you can view your permissions on tables, files, and APIs respectively.

2.12.6. Perform cross-tenant authorization

Authorization logic

The logic of authorization within a tenant and that of cross-tenant authorization are as follows:

- Authorization within a tenant: An access control list (ACL) is used for authorization. You need to add the applicant to the corresponding workspace and then grant permissions to the applicant as requested.

- Cross-tenant authorization: A package is used for authorization. First, check whether the workspace where the requested resources reside has a package.
  - If the workspace does not have a package, create a package and add the requested resources to the package. Then, install the package in the workspace where the requested resources are used.

    After that, use an ACL to grant permissions to the applicant as requested.
  - If the workspace has a package, add the requested resources to the package.

  If the requested resources need to be shared with multiple workspaces, install the package in all these workspaces. If the workspace where the requested resources reside has multiple packages, install all of them in the workspaces where the requested resources are used.

Procedure

1. Log on to the DataWorks console. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Asset Management**.
2. On the **Asset Portal** tab, enter the name of the table that belongs to another tenant in the search box and click **Search**.
3. On the **Asset Category** tab that appears, click the name of the found table on the **Tables** tab. The details page of the table appears.

   On the details page, you can view details about the table, including the basic information, business information, physical information, and field information.
4. Click **Request Permission** in the upper-right corner. The **Table Permission Request** page appears.
5. Set parameters on the **Table Permission Request** page.

Parameter	Description
Target Workspace	The workspace where the table is to be used.
Environment	The type of the environment where the table is to be used. If the selected workspace is in standard mode, you need to set this parameter to Development or Production.
Home Workspace	The workspace where the table resides. This parameter is automatically set.
Home Tenant	The tenant to which the table belongs. This parameter is automatically set.
Grant To	The account for which permissions on the table are requested. Valid values: **Current Account** and **System Account for Production Environment**.
Requested Period	The period in which the requested permissions are valid.

Parameter	Description
Reason for Request	The reason why you request the permissions. To improve the review efficiency, we recommend that you describe the reason in detail.
Objects Requested	The tables you want to use. By default, the current table is selected. You can select other tables.

6. After the configuration is completed, click **Submit**.

   After you submit the request, click **Approval Management** in the top navigation bar and then click **Submitted by Me**. You can view the review progress on the page that appears.

   - If you want to revoke a request, click **Revoke** in the Actions column of the request.
   - If you want to submit a revoked or rejected request again, click **Reapply** in the Actions column of the request.

7. Log on to the DataWorks console as the table owner, go to the **Approval Management** tab and click **To Be Handled**. On the page that appears, review the request information and click **Agree**.

   After clicking **Agree**, click **Approval Management** in the top navigation bar and click **Handled by Me**. On the page that appears, you can view your authorization records and revoke specific authorizations.

2.13. Organization management

2.13.1. Member management

1. Log on to the DataWorks console as a workspace administrator.
2. Move your pointer over the DataWorks icon in the upper-left corner, and click **Project Management**.
3. Click **Member Management** in the left-side navigation bar. The **Members** page appears.
4. You can enter a member name or logon name in the search box to search for a member to be added or removed from the current organization.
   - Assign a role

     To assign a role to a member, click the **Roles** drop-down list next to the member, and select the role to be assigned.

     To unassign a role from a member, click **x** next to the role.
   - Remove a member from an organization

     Click **Delete** next to the member, and click **OK** in the **Remove from Tenant** dialog box that appears.

2.13.2. Resource groups

2.13.2.1. About scheduling resources

You can use the Scheduling Resource page to create, configure, and edit a scheduling resource.

A scheduling resource is an object within an organization. A dedicated scheduling resource may contain multiple physical machines or ECS instances that are used to implement a specific task.

1. Log on to the DataWorks console.
2. In the left-side navigation pane, choose **Organization Management > Scheduling Resources**.

> **Note** On the **Scheduling Resources** page, the tenant administrator can create a dedicated scheduling resource, and edit an existing scheduling resource.

2.13.2.2. Change the workspace of scheduling resources

You can change the workspace of dedicated scheduling resources that have been created and configured.

Procedure

To change the workspace of dedicated scheduling resources, the tenant administrator performs the following operations:

1. Log on to the DataWorks console as a tenant administrator.
2. Choose **Organization Management > Scheduling Resources**.
3. On the page that appears, enter a scheduling resource name for a fuzzy search to find the target scheduling resource.
4. Click **Change Workspace**.
5. Click **OK**.

2.13.3. Configure the compute engine

Currently, DataWorks only supports MaxCompute as its compute engine. All business flows and nodes in a workspace are run on the MaxCompute project associated to the workspace.

Example

> **Note** Tenant administrators can modify the settings for MaxCompute projects. The following settings are changeable: the project description, whether to use the MaxCompute project owner account to run MaxCompute jobs, the account used for running MaxCompute jobs, and the AccessKey of the account.

Assume that the account used for running MaxCompute jobs is no longer available, for example, because the account owner has resigned. If Run MaxCompute Task Using MaxCompute Owner Account is not selected, the tenant administrator needs to immediately modify the account used for running MaxCompute jobs and its AccessKey so that tasks can properly run in the workspace that uses the corresponding MaxCompute project.

Procedure

You can modify the account used for running MaxCompute jobs and its AccessKey as follows:

1. Log on to the DataWorks console as a tenant administrator. For more information, see Log

on to the DataWorks console.

2. Choose **Project Management** > **Compute Engine**.
3. In the search box on the **Project Management** > **Compute Engine** page, enter the compute engine name. Fuzzy search is supported.
4. Find the target compute engine, and click **Configure** in the Actions column.
5. In the **Configure Compute Engine** dialog box, specify the Alibaba Cloud Account and the AccessKey.

   > **Note** You can also select Run MaxCompute Task Using MaxCompute Owner Account or create a new Alibaba Cloud account.

6. Click **Submit**.

2.14. Data Service

2.14.1. Overview

With Data Service, you can manage all your table APIs after you create new APIs or register existing APIs. You can also easily publish your APIs to API Gateway. Together with API Gateway, Data Service provides a secure, stable, low-cost, and easy-to-use data sharing service.

Data Service adopts a serverless architecture and allows you to develop table APIs without thinking about infrastructure such as compute resources. Data Service supports automatic scaling for compute resources, which significantly reduces your OPEX.

Create an API

In Data Service, you can quickly create APIs based on tables in relational databases or NoSQL databases using a visual wizard. It takes only a few minutes to configure a data API, and coding is not required. You can also create APIs by specifying SQL scripts. The script mode supports advanced functions such as associative tables, complex criteria, and aggregate functions.

Register an API

You can register existing RESTful APIs to Data Service for unified API management. Four request methods and three data formats are supported. The four request methods are GET, POST, PUT, and DELETE. The three data formats are tables, JSON, and XML.

API Gateway

API Gateway provides API lifecycle management services, including API publishing, management, maintenance, and monetization. It enables low-risk, simple, cost-effective, and fast microservice integration, front and back end separation, and system integration. You can use API Gateway to share functions and data with your partners and third-party developers.

API Gateway supports authorization, authentication, flow control, and billing for Data Service.

2.14.2. Terms

This section introduces terms of Data Service.

Name	Description
Data source	Indicates database links. Data Service accesses data through data sources. Data sources are configured in Data Integration.
Create an API	Creates APIs based on data tables.
Register an API	Registers existing APIs to Data Service for unified management.
Wizard mode	Guides you through the procedure of API creation. This method is suitable for beginners who want to create simple APIs. You do not need to write any code.
Script mode	Allows you to create APIs by writing SQL scripts. This method supports associative tables, complex queries, and aggregate functions. This method is suitable for experienced developers who want to create complex APIs.
API group	Indicates a set of APIs for a specific scenario or for consuming a specific service. An API group is the smallest group unit in Data Service, and the smallest unit for API Gateway management. API groups are published in Alibaba Cloud API Marketplace as API products.
API Gateway	Indicates a hosted service provided by Alibaba Cloud to manage APIs. API Gateway supports API lifecycle management, permission management, access management, and traffic control.

2.14.3. Manage tags

This topic describes how to create, add, view, and remove tags for an API.

DataService Studio allows you to add tags to APIs when you manage workflows and create, register, and deploy APIs. This topic uses the scenario of creating an API in the codeless UI as an example. Tags allow you to efficiently classify and search for APIs. You can maintain only one tag list in a workspace, and cannot use the tag list of a workspace in another workspace.

> **Note**
>
> - Each API supports zero to five tags. That is, you can add no tags to an API at all or add a maximum of five tags to an API.
> - The name of a tag can contain letters, digits, and underscores (_), and cannot exceed 20 characters in length.

Create a tag for an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products > Data Service**.
2. On the Service Development tab, move the pointer over  and choose **API > Generate API**.
3. In the **Generate API** dialog box, set the API mode parameter to Wizard Mode and enter a tag in the **Label** field.

If the tag you entered does not exist in the current workspace, Add <Tag> appears in the drop-down list. Click Add <Tag> to create the tag for the API.

4. Set other parameters and click **OK**. The tag is created for the API and appears in the tag list of the current workspace.

Add an existing tag to an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products** > **Data Service**.
2. On the Service Development tab, move the pointer over  and choose **API** > **Generate API**.
3. In the **Generate API** dialog box, set the API mode parameter to Wizard Mode and click a blank area or the downward arrow in the **Label** field. Tags in the current workspace appear in a drop-down list.
4. Click the required tag, set other parameters, and then click **OK**. The tag is added to the API.

View tags of an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products** > **Data Service**.
2. On the **Service Development** tab, double-click the name of the target API in the API list.
3. In the right-side navigation pane, click **Properties**. Then you can view the tags of the API in the **Label** column.

   > **Note** You can also create, add, and remove tags for the API in the Properties pane.

   If an API is published, you can also perform the following steps to view the tags of the API:

   i. In DataService Studio, click **Service Management** in the upper-right corner.

   ii. Click **Manage APIs** in the left-side navigation pane. On the page that appears, click the **APIs of Published** tab and click the name of the target API. The **API Details** page appears.

   iii. View the tags of the API under **Label** in the **API Basic Information** section.

Remove a tag from an API

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products** > **Data Service**.
2. On the **Service Development** tab, double-click the name of the target API in the API list.
3. In the right-side navigation pane, click **Properties**. Then, you can view the tags of the API in the **Label** column.
4. Click  next to the tag to remove.
5. Click **Publish** in the upper-right corner. The tag is removed from the API.

Search for APIs by tag

1. Log on to the DataWorks console, click the DataWorks icon in the upper-left corner, and then choose **All Products** > **Data Service**.

2. On the **Service Development** tab, click **Service Management** in the upper-right corner.
3. Click **Manage APIs** in the left-side navigation pane. On the page that appears, click the **APIs of Published** tab and view the tags of each API in the **Label** column.

   If an API has multiple tags and some of them are hidden, click **...** to show all the tags.
4. On the **APIs of Published** tab, click **Advanced Search**.
5. Enter a tag in the **Label** field to search for all APIs associated with the tag.

   > **Note** You can search for APIs based on multiple tags.

2.14.4. Manage business processes and objects under business processes

2.14.4.1. Manage business processes

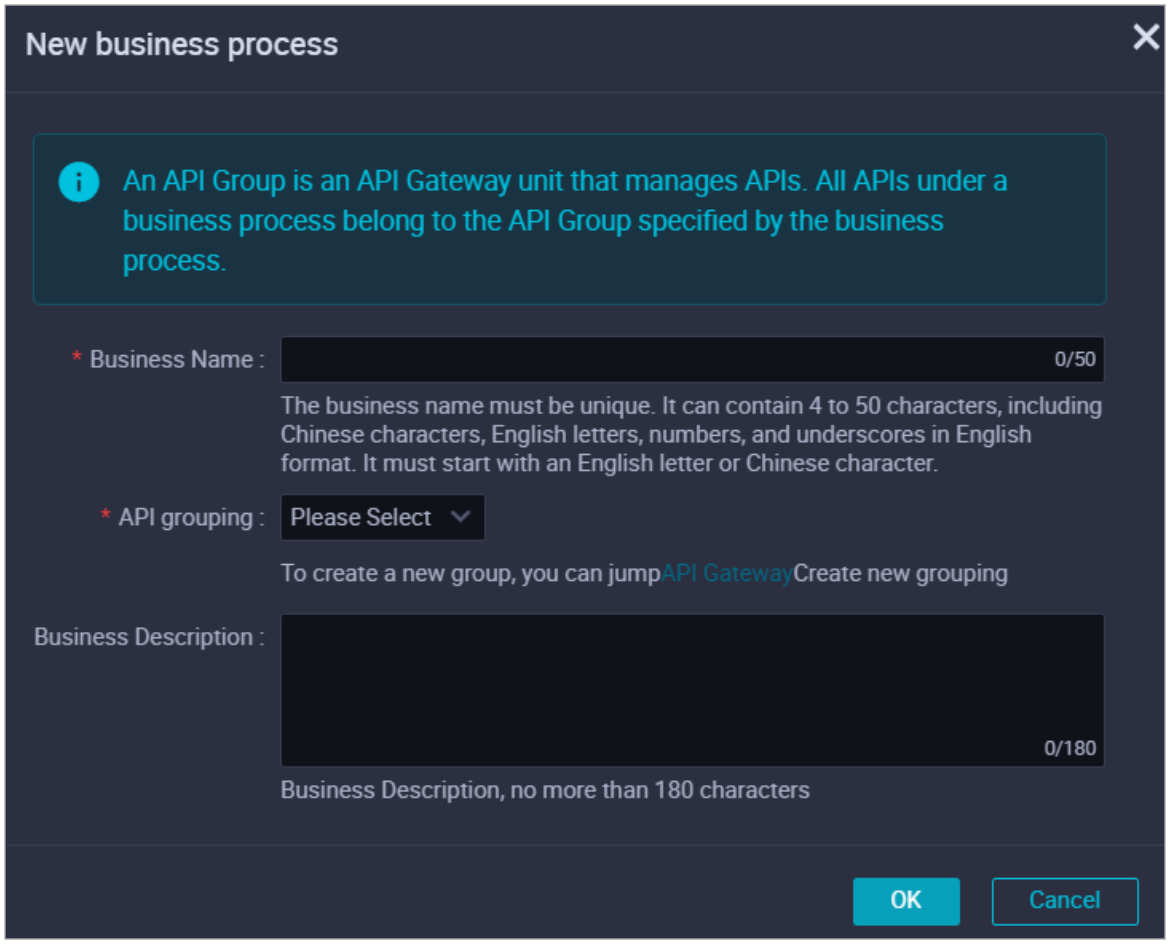
This topic describes how to create, modify, and delete a business process.

Context

DataWorks allows you to organize different types of resources in a business process. This helps you analyze data by business. Each business process contains APIs, functions, and workflows.

Create a business process

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over  and select **Business process**.
4. In the **New business process** dialog box, set the parameters as required.

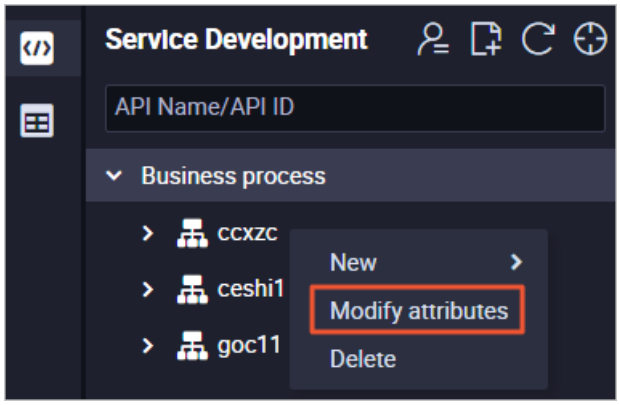


Parameter	Description
Business Name	The name of the business process. ○ The name can contain letters, digits, and underscores (_). ○ The name must start with a letter. ○ The name must be 4 to 50 characters in length. ○ The name must be unique in the workspace to which the business process belongs.
API Group	The API group to which the APIs under the business process belong. An API group is the API management unit of API Gateway. You can select an API group from the drop-down list. DataService Studio creates a default API group named in the Workspace name_default format for each user. If you need to create an API group, click **API Gateway** in the note to go to the API Gateway console and create an API group.
Business Description	The description of the business process. The description can be up to 180 characters in length.

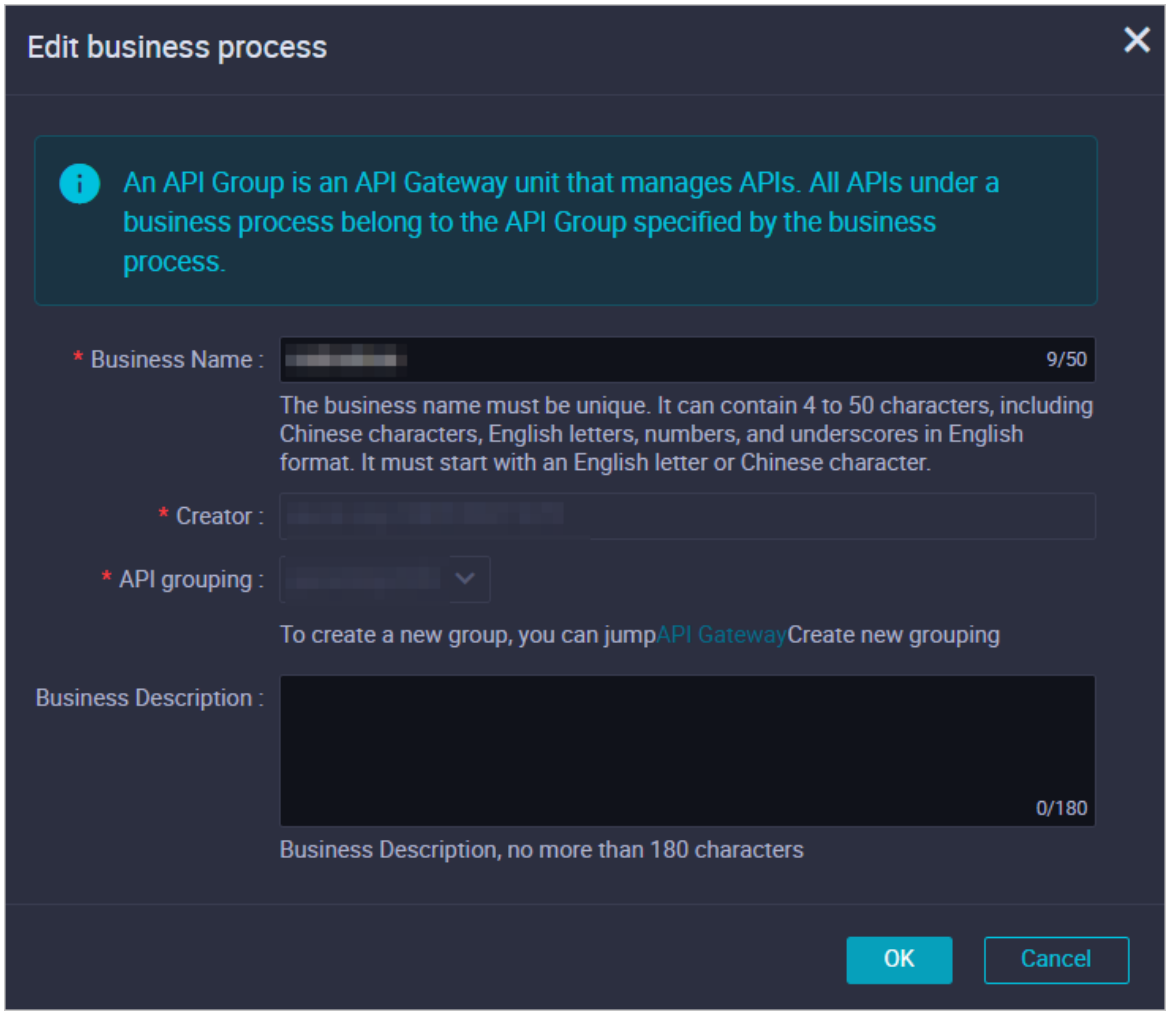
5. Click **OK**.After the business process is created, you can view it in the business process list.

Modify a business process

1. On the **Service Development** tab, right-click the name of the target business process and select **Modify attributes**.



2. In the **Edit business process** dialog box, modify the **Business Name** and **Business Description** parameters as required.

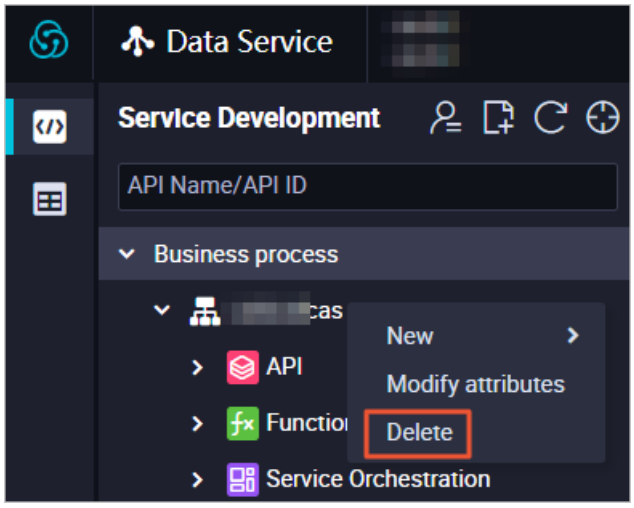


> **Note** You cannot modify the **Creator** or **API Group** parameter of a business process.

3. Click **OK**.

Delete a business process

1. On the **Service Development** tab, right-click the name of the target business process and select **Delete**.



2. In the message that appears, click **OK**.

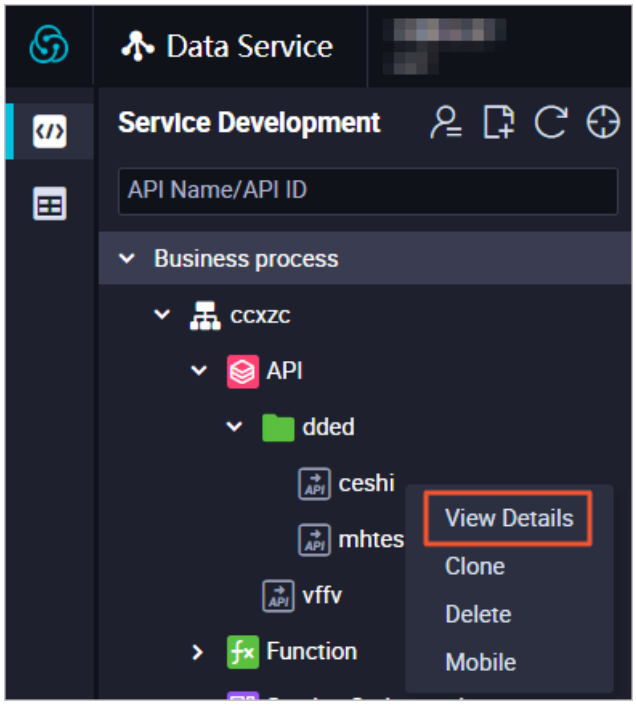
> **Note**
> - You can delete only business processes that do not contain any objects such as APIs, functions, or workflows.
> - If you need to delete a business process that contains such objects, delete the objects before you delete the business process.

2.14.4.2. Manage APIs

This topic describes how to view, clone, delete, and move APIs.

View an API

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, right-click the name of the target API and select **View Details**.

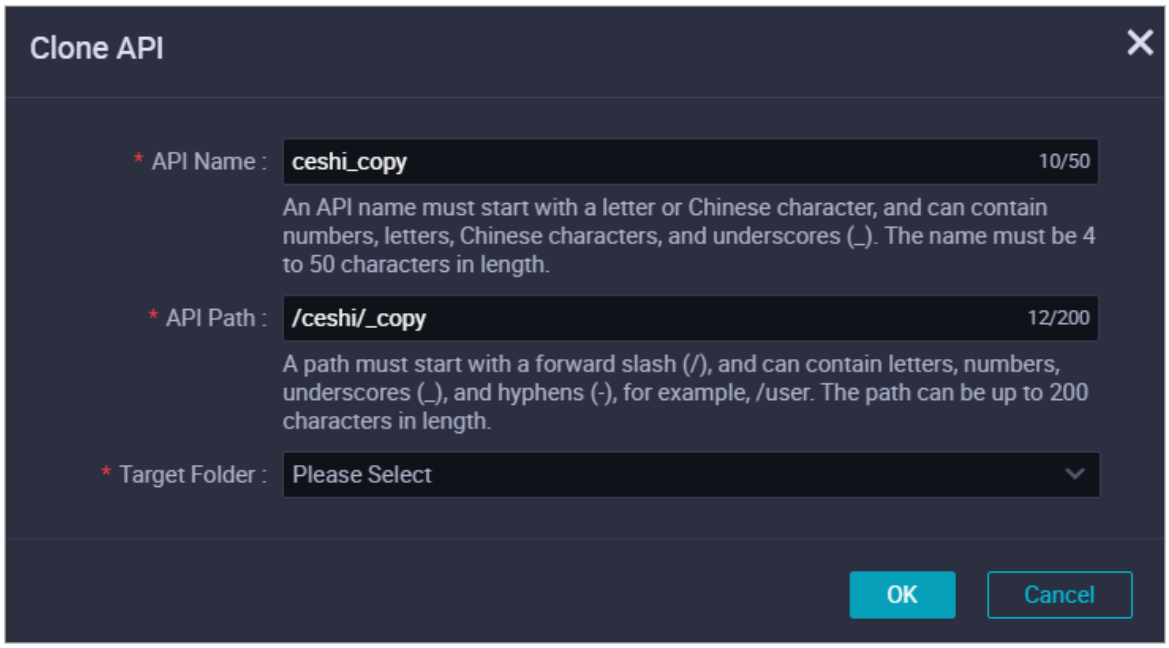


Note The View Details option appears only in the shortcut menu of an API that has been published. If an API has not been published, double-click the API to go to the configuration tab of the API. Then, click Properties in the right-side navigation pane to view its basic information.

Clone an API

You can clone an API to a specified directory in the directory tree.

1. On the **Service Development** tab, right-click the name of the target API and select **Clone**.
2. In the **Clone API** dialog box, set the parameters as required.



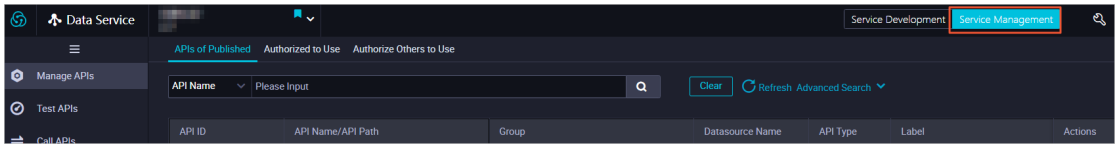
Parameter	Description
API Name	The name of the cloned API. It must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the cloned API, for example, */user*. The path can contain letters, digits, underscores (_), and hyphens (-). It must start with a forward slash (/) and can be up to 200 characters in length.
Target Folder	The directory for storing the cloned API.

3. Click **OK**.

Delete an API

You can delete only APIs that have not been published. To delete APIs that have been published, you must unpublish them first.

1. Optional. Unpublish the target API. If the API to be deleted is in the Unpublished state, skip this step.
   i. Go to the **Service Development** tab and click **Service Management** in the upper-right corner.



   ii. On the page that appears, click the **APIs of Published** tab, find the target API, and then click **Unpublish** in the Actions column.
   iii. In the **Unpublish API** message, click **OK**.
   iv. Click **Service Development** in the upper-right corner to return to the **Service Development** tab.
2. On the **Service Development** tab, right-click the name of the target API and select **Delete**.
3. In the **Delete API** message, click **OK**.

> **Note** Deleted APIs cannot be recovered. Use caution when you delete an API.

Move an API to another directory

You can move only APIs that have not been published. To move APIs that have been published, you must unpublish them first.

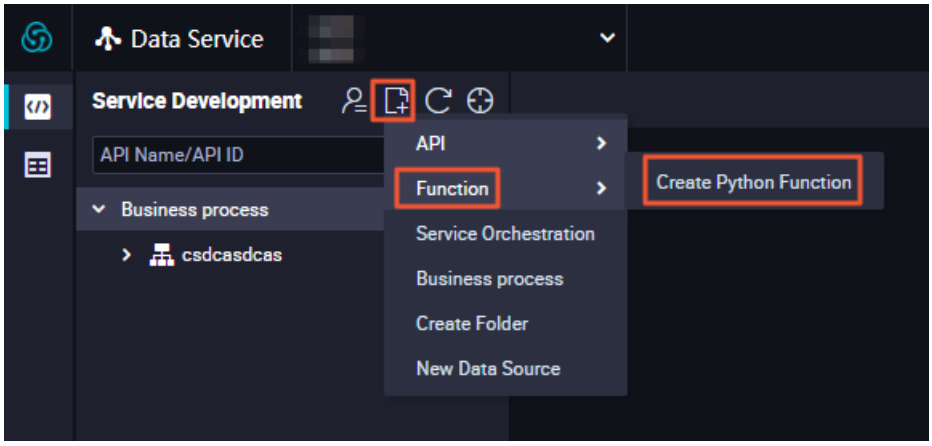
1. On the **Service Development** tab, right-click the name of the target API and select **Mobile**.
2. In the **Modify file path** dialog box, set the **Target Folder** parameter.
3. Click **OK**.

2.14.4.3. Manage functions

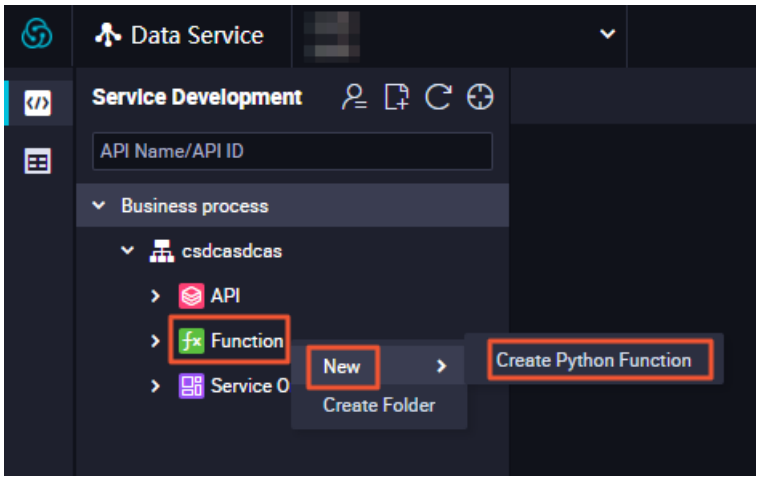
This topic describes how to create, clone, delete, and move Python functions.

Create a function

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. Move the pointer over and choose **Function > Create Python Function**.

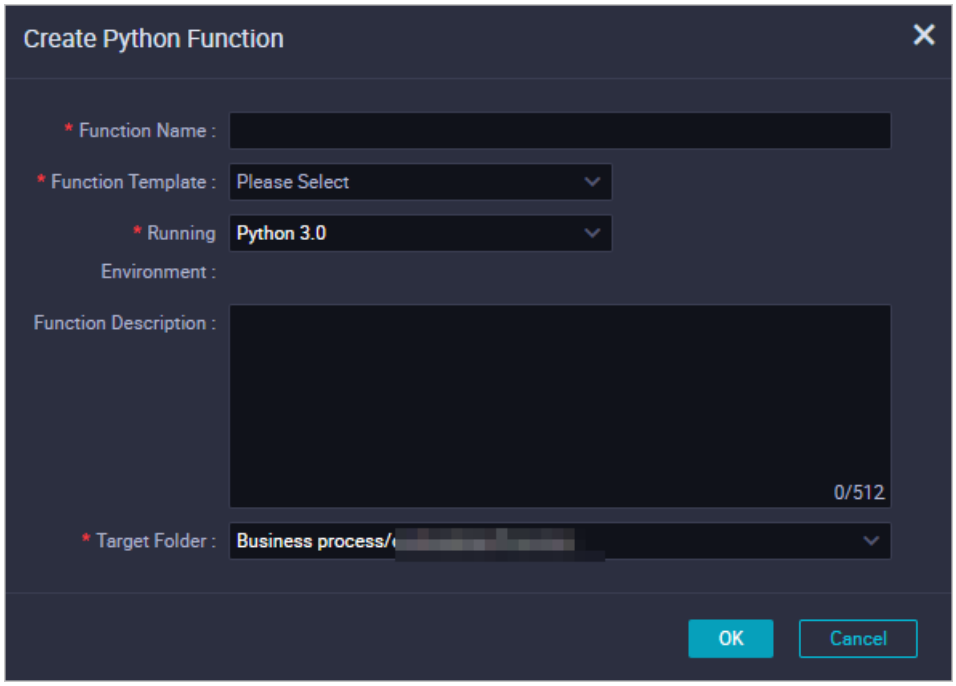


You can also click a business process, right-click **Function**, and then choose **New > Create Python Function**.



> **Notice** DataService Studio allows you to create only Python functions.

4. In the **Create Python Function** dialog box, set the parameters as required.

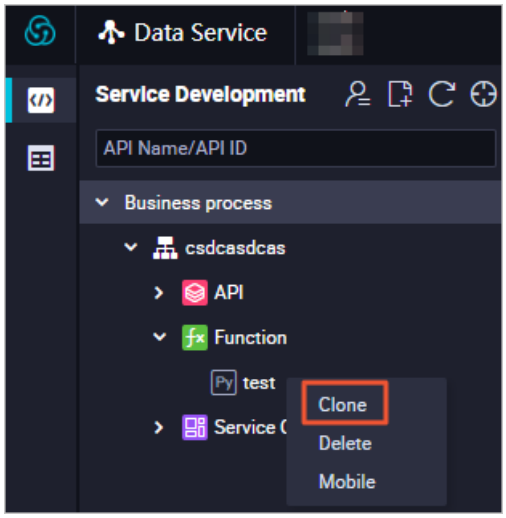


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function. The description can be up to 512 characters in length.
Target Folder	The directory for storing the function.

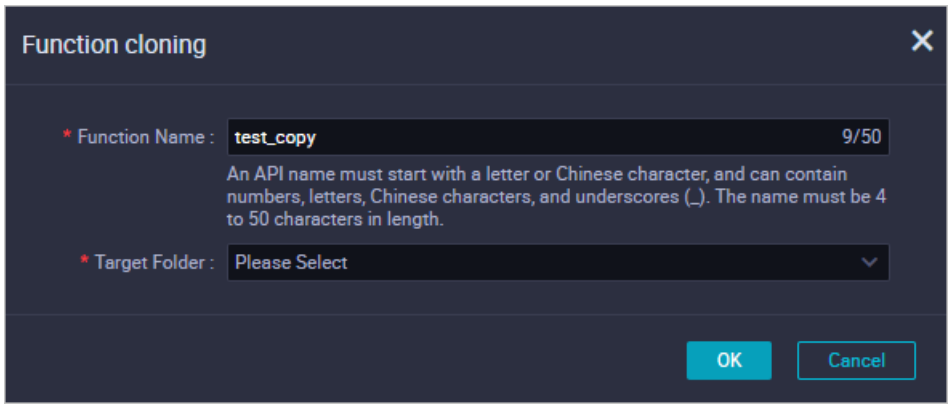
5. Click **OK**.
6. On the configuration tab of the function, configure the function.
   i. In the **Edit Code** section, enter the function code.
   ii. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.
7. Click Save icon in the toolbar.

Clone a function

1. On the **Service Development** tab, right-click the name of the target function and select **Clone**.



2. In the **Function cloning** dialog box, set the **Function Name** and **Target Folder** parameters.

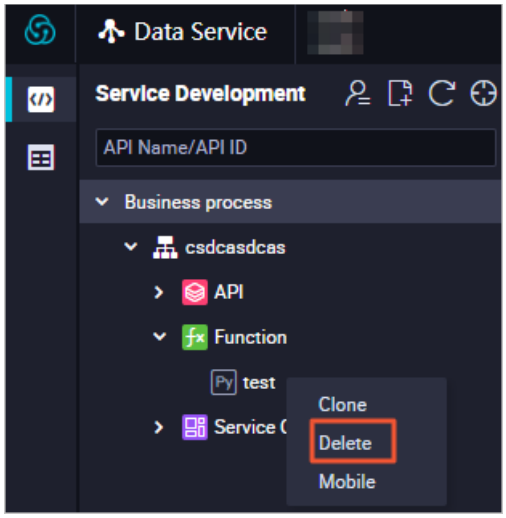


> **Note** The name of the function must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.

3. Click **OK**.

Delete a function

1. On the **Service Development** tab, right-click the name of the target function and select **Delete**.

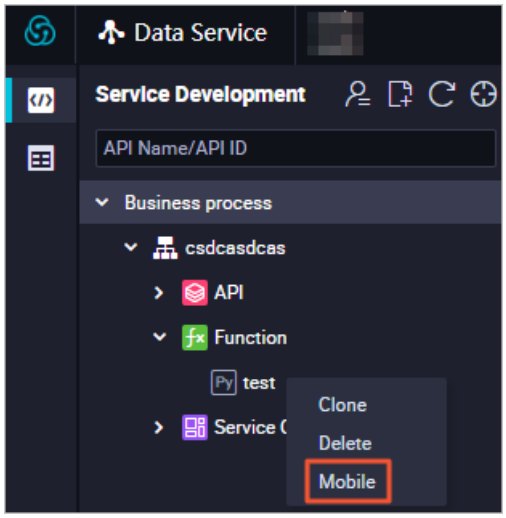


2. In the message that appears, click **OK**.

> **Note** You can delete only functions that are not referenced by APIs. You must remove the function from the filters of the APIs that reference the function before you can delete the function.

Move a function to another directory

1. On the **Service Development** tab, right-click the name of the target function and select **Mobile**.



2. In the **Modify file path** dialog box, set the **Target Folder** parameter.
3. Click **OK**.

2.14.4.4. Manage workflows

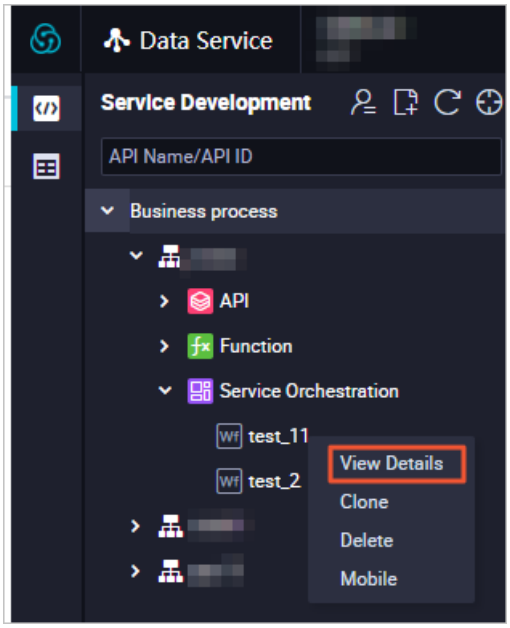
This topic describes how to view, clone, delete, and move a workflow.

Prerequisites

Workflows are created and published. For more information, see Use workflows.

View a workflow

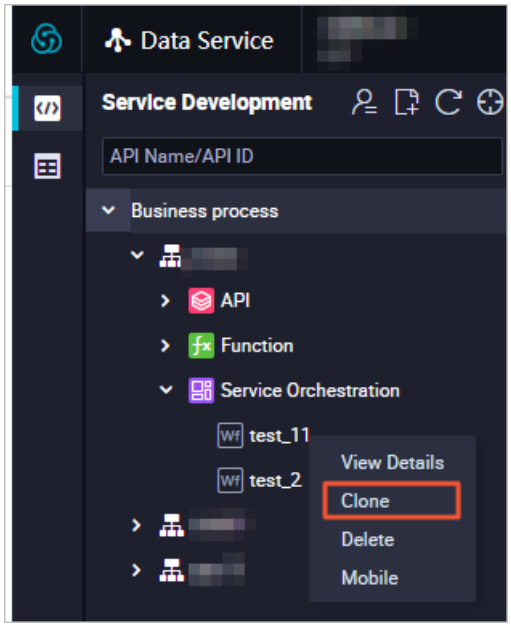
1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the Service Development tab, right-click the name of the target workflow and select **View Details**.



> **Note** The View Details option appears only in the shortcut menu of a workflow that has been published. If a workflow has not been published, double-click the workflow to go to the configuration tab of the workflow. Then, click **Properties** in the right-side navigation pane to view its basic information.

Clone a workflow

1. On the **Service Development** tab, right-click the name of the target workflow and select **Clone**.



2. In the **Clone API** dialog box, set the parameters as required.

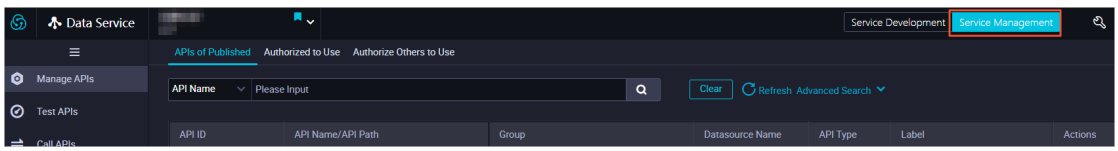
Parameter	Description
API Name	The name of the cloned workflow. It must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the cloned workflow, for example, */user*. The path can contain letters, digits, underscores (_), and hyphens (-). It must start with a forward slash (/) and can be up to 200 characters in length.
Target Folder	The directory for storing the cloned workflow.

3. Click **OK**.

Delete a workflow

You can delete only workflows that have not been published. To delete workflows that have been published, you must unpublish them first.

1. Optional. Unpublish the target workflow. If the workflow to be deleted is in the Unpublished state, skip this step.
   i. Go to the **Service Development** tab and click **Service Management** in the upper-right corner.



   ii. On the page that appears, click the **APIs of Published** tab, find the target API, and then click **Unpublish** in the Actions column.
   iii. In the **Unpublish API** message, click **OK**.

iv. Click **Service Development** in the upper-right corner to return to the **Service Development** tab.

2. On the **Service Development** tab, right-click the name of the target workflow and select **Delete**.
3. In the **Delete API** message, click **OK**.

> **Note** Deleted workflows cannot be recovered. Use caution when you delete a workflow.

Move a workflow to another directory

You can move only workflows that have not been published. To move workflows that have been published, you must unpublish them first.

1. On the **Service Development** tab, right-click the name of the target workflow and select **Mobile**.
2. In the **Modify file path** dialog box, set the **Target Folder** parameter.
3. Click **OK**.

2.14.5. Create an API

In Data Service, you can quickly create APIs based on tables in relational databases or NoSQL databases using a visual wizard. It takes only a few minutes to configure a data API, and coding is not required.

You can also create APIs by specifying SQL scripts. The script mode supports advanced functions such as associative tables, complex criteria, and aggregate functions.

The differences between the wizard mode and script mode are described as follows:

Differences between the wizard mode and script mode

Category	Description	Wizard mode	Script mode
Query object	Queries a single table from one data source	Supported	Supported
	Queries associative tables from one data source	Not supported	Supported
Search condition	Searches for an exact number	Supported	Supported
	Searches for a range of numbers	Not supported	Supported
	Matches an exact string	Supported	Supported
	Performs fuzzy search for strings	Supported	Supported

Category	Description	Wizard mode	Script mode
	Sets required and optional parameters	Supported	Supported
Query result	Returns the field value	Supported	Supported
	Performs a mathematical calculation for field values	Not supported	Supported
	Performs an aggregate operation on field values	Not supported	Supported
	Displays results with pagination	Supported	Supported

2.14.5.1. Configure connections

DataService Studio allows you to obtain table schemas and query data through APIs from connections.

> **Note** Before generating an API, make sure that you have configured the relevant connections.

You can click **Connections** on the **Data Integration** page to configure connections. The following table lists the available connection types and supported configuration modes.

Connection name	Create an API in the codeless UI	Create an API in the code editor
ApsaraDB for RDS	Supported	Supported
DRDS	Supported	Supported
ApsaraDB for MySQL	Supported	Supported
ApsaraDB for PostgreSQL	Supported	Supported
ApsaraDB for SQL Server	Supported	Supported
Oracle	Supported	Supported
AnalyticDB	Supported	Supported
Table Store	Supported	Not supported
MongoDB	Supported	Not supported

2.14.5.2. Create an API in the codeless UI

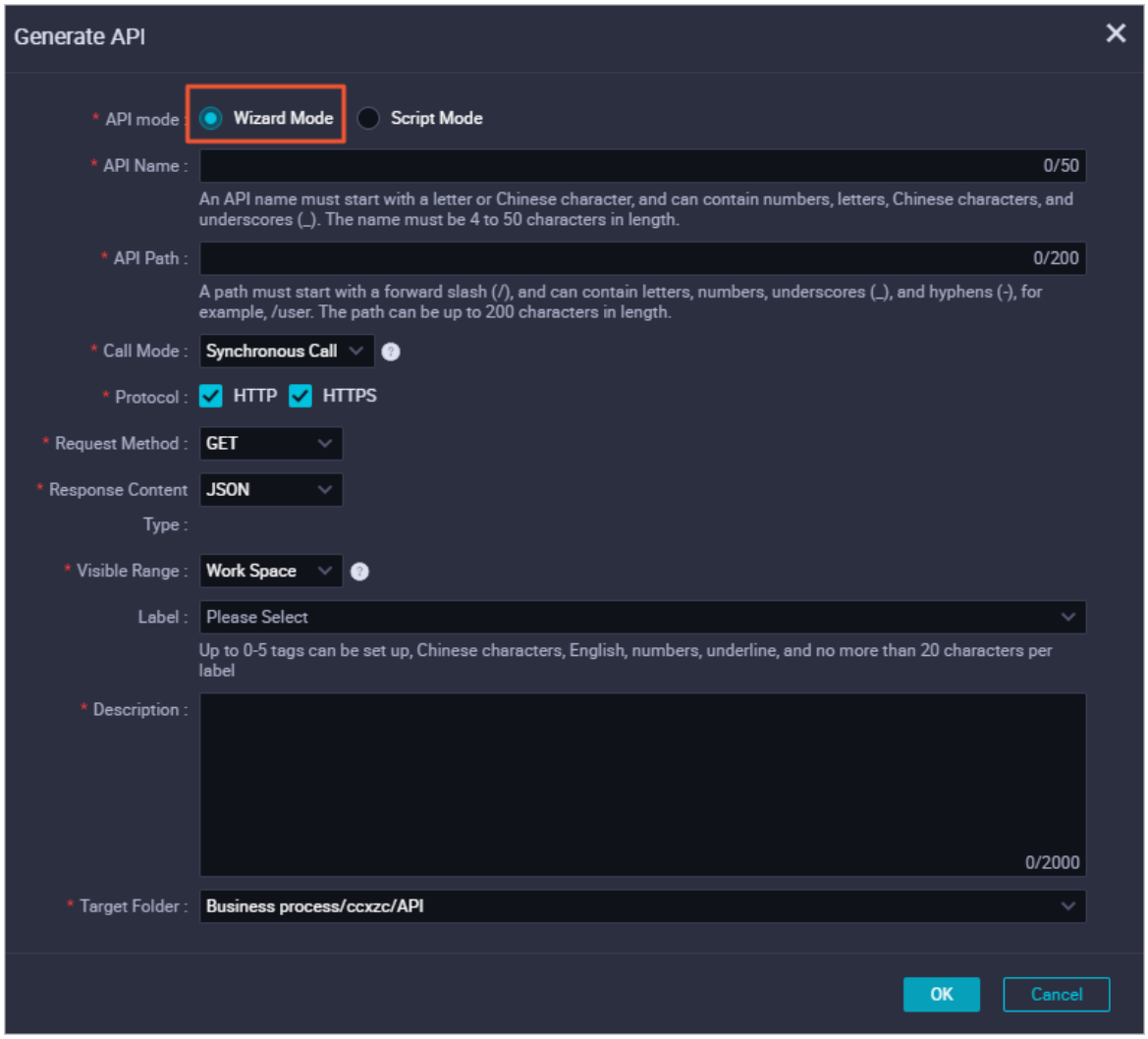
DataWorks allows you to create APIs by setting parameters in the codeless UI without the need to write code. This topic describes how to create an API in the codeless UI.

Prerequisites

Connections are configured on the **Data Source** page. For more information, see Configure data sources.

Create an API

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over and choose **API > Generate API**.You can also click a business process, right-click **API**, and then choose **New > Generate API**.
4. In the **Generate API** dialog box, set the parameters as required.



Parameter	Description
API mode	The mode for creating the API. Valid values: **Wizard Mode** and **Script Mode**. In this example, select **Wizard Mode**.
API Name	The name of the API. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the API, for example, */user*.
Call Mode	The mode for calling the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Mode**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Mode**, the API returns the RequestID parameter immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET** and **POST**.
Response Content Type	The return type of the API. Set the value to **JSON**.
Visible Range	The visibility of the API. Valid values: ○ **Work Space**: The API is visible to all members in the current workspace. ○ **Private**: The API is visible only to its owner and permissions on the API cannot be granted to other users. **Note** If you set the Visible Range parameter to Private, the API is visible only to you in the directory tree. It is hidden to other members of the workspace.
Label	The tag of the API. Select one or more tags from the drop-down list. For more information, see Manage tags. **Note** You can set at most five tags for an API.
Description	The description of the API, which can be up to 2,000 characters in length.
Target Folder	The directory for storing the API.

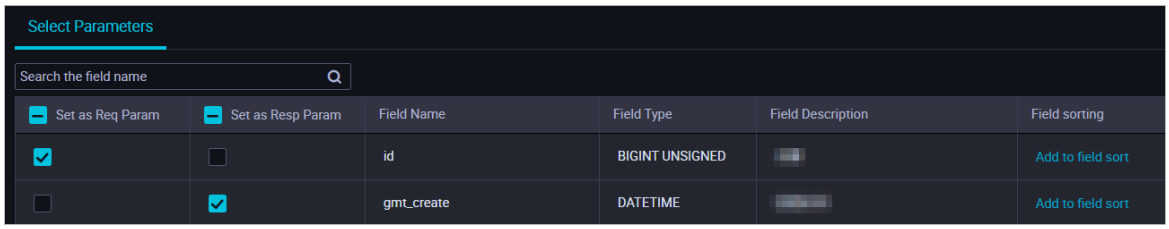
5. Click **OK**.

Configure the API

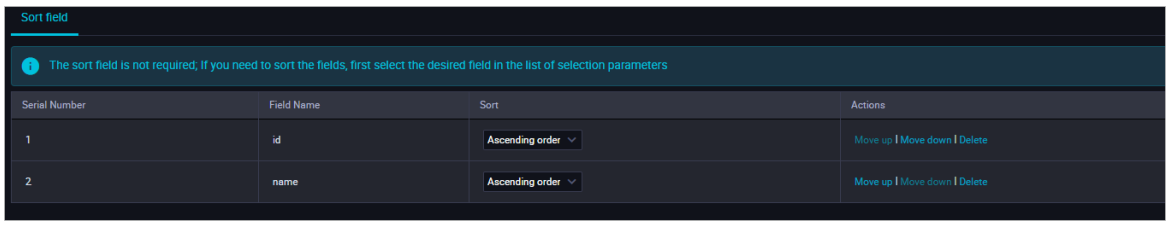
1. Double-click the API in the directory tree. On the configuration tab that appears, set the **Datasource Type**, **Datasource Name**, and **Table Name** parameters in the **Select Table** section.

   > **Note**
   > - Before you select a table for an API, you must configure a connection in Data Integration. You can enter a table name in the Table Name field to search for the desired table.
   > - After you create an API, the table configuration tab automatically appears for you to select a table for the API.

2. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.
3. In the **Select Parameters** section, set the request and response parameters for the API.After you select a table in the Select Table section, all fields in the table appear in the **Select Parameters** section. Select the required fields and select the check boxes in the **Set as Req Param** and **Set as Resp Param** columns as required.

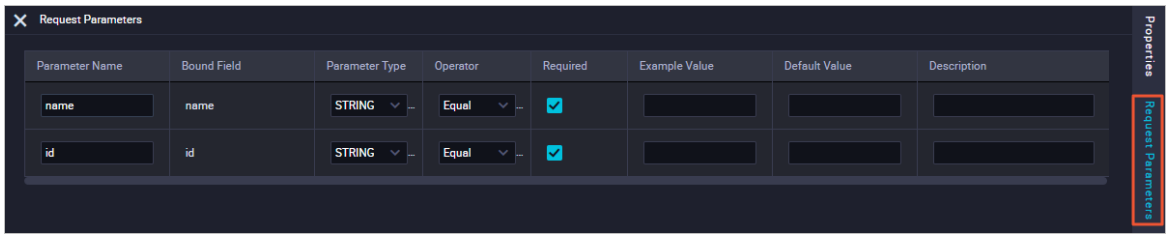


   To sort the data returned by the API based on a field, click **Add to field sort** in the Actions column of the field to add it to the **Sort field** section.



   The sorting feature allows you to specify the fields based on which the parameters returned by the API are sorted. A field with a smaller sequence number in the Sort field section has a higher priority in sorting. You can click **Move up** or **Move down** to adjust the sequence of a field. You can specify the sorting mode for each field by selecting **Ascending order** or **Descending order** in the Sort column.

4. In the right-side navigation pane, click **Request Parameters**. In the Request Parameters pane, set the parameters as required.

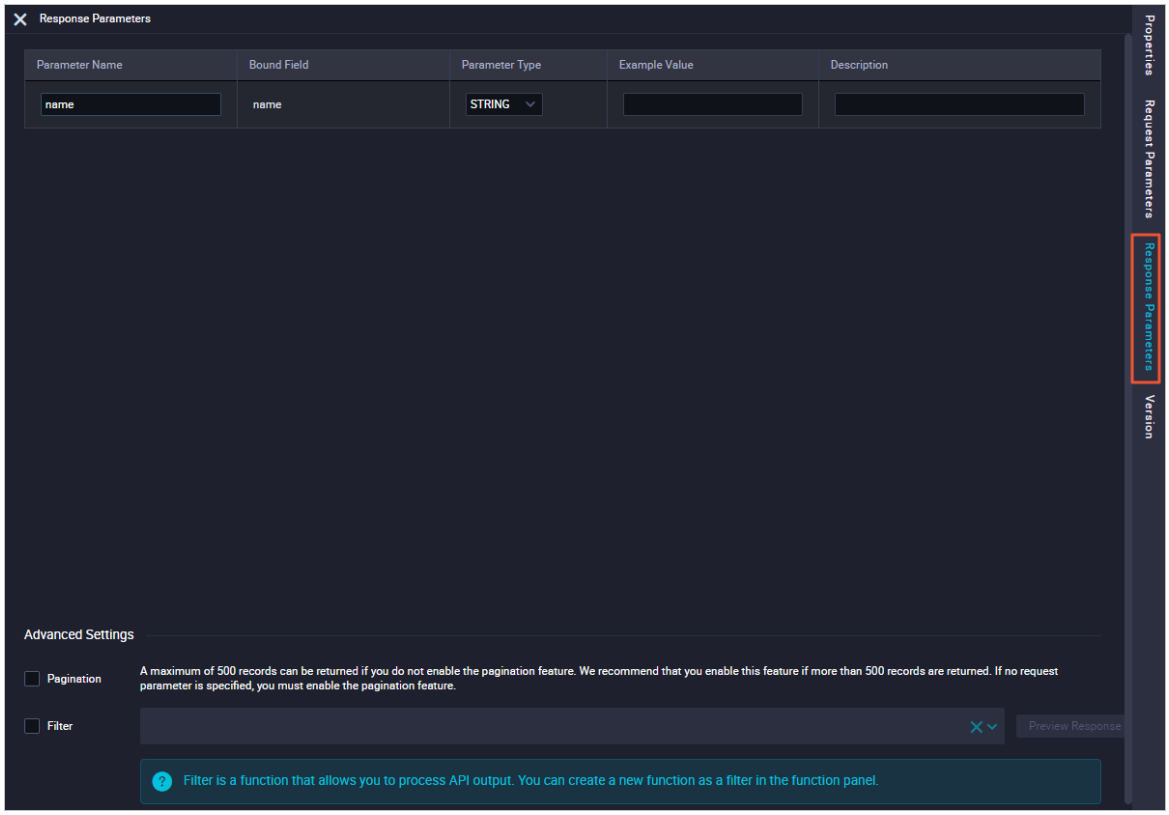


Parameter	Description
Parameter Name	The name of the request parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Bound Field	The field to be bound to the request parameter. You cannot change the value.
Parameter Type	The type of the request parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Operator	The operator that is used to associate or compare the value of the request parameter with the specified value. You can select one of the following operators: ○ **Equal**: The value of the request parameter is equal to the specified value. ○ **LIKE**: The value of the request parameter matches the specified pattern. ○ **IN**: The value of the request parameter is in the specified range. ○ **NOT IN**: The value of the request parameter is out of the specified range. ○ **NOT LIKE**: The value of the request parameter does not match the specified pattern. ○ **! =**: The value of the request parameter is not equal to the specified value. ○ **>**: The value of the request parameter is greater than the specified value. ○ **<**: The value of the request parameter is less than the specified value. ○ **>=**: The value of the request parameter is greater than or equal to the specified value. ○ **<=**: The value of the request parameter is less than or equal to the specified value.
Required	Specifies whether the request parameter is required.
Example Value	The sample value of the request parameter.
Default Value	The default value of the request parameter.
Description	The description of the request parameter.

To preprocess the request parameters of the API, select **Use prefilter** in the **Advanced Settings** section. For more information, see Use prefilters.

> **Note**
>
> - To enhance the matching efficiency, set an indexed field as a request parameter.
> - To make it easier for API callers to know the details about the API, we recommend that you specify information such as the sample value, default value, and description for each parameter of the API.

5. In the right-side navigation pane, click **Response Parameters**. In the Response Parameters pane, set the parameters as required.



Parameter	Description
Parameter Name	The name of the response parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Bound Field	The field to be bound to the response parameter. You cannot change the value.
Parameter Type	The type of the response parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Example Value	The sample value of the response parameter.
Description	The description of the response parameter.

You can select **Pagination** and **Filter** in the **Advanced Settings** section.

Select Pagination based on your needs.

- If you do not select **Pagination**, the API returns a maximum of 2,000 records by default.
- If the API may return more than 2,000 records, we recommend that you select **Pagination**.

The following common parameters are available when Pagination is selected:

- Common request parameters
  - pageNum: the number of the page to return.
  - pageSize: the number of entries to return on each page.
- Common response parameters
  - pageNum: the page number of the returned page.
  - pageSize: the number of entries returned per page.
  - totalNum: the total number of returned entries.

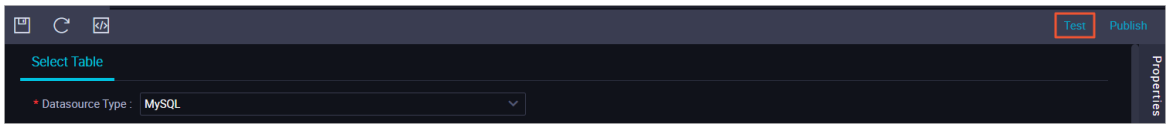
If you need to process the query results returned by the API, select **Filter**.

> **Note**
>
> - Field values are returned in the response as they are in the table.
> - Request parameters are optional for an API. If you do not specify any request parameters for an API, you must select **Pagination**.

6. Click Save icon in the toolbar.

Test the API

1. After you save the settings of the API, click **Test** in the upper-right corner.



2. In the **Test APIs** dialog box, click **Test** to send an API request.The request and response details appear on the right. If the API fails the test, check the error message, modify the API settings accordingly, and test the API again.

   You can select **Save the correct response example automatically** as required.

> **Note**
>
> - The system automatically generates sample failure responses and error codes when it tests an API. However, the system does not automatically generate sample success responses.
>
> To allow the system to save the success test result as a sample success response, you must select **Save the correct response example automatically** before you perform the test. If the response contains sensitive data that must be de-identified, you can manually edit the response.
> - The sample success response is an important reference for API callers, and therefore must be configured.
> - The Call Latency value is the latency of the current API request, which is used to evaluate the API performance. If the latency is long, consider optimizing the database.

3. After the API is tested, close the **Test APIs** dialog box and click **Publish** in the upper-right corner of the configuration tab.

Switch from the codeless UI to the code editor

On the configuration tab of an API, you can switch from the codeless UI to the code editor.

1. Go to the **Service Development** tab and double-click the target API. The configuration tab of the API appears.
2. Click in the toolbar.
3. In the message that appears, click **OK**. Then, you can view the SQL statements of the API in the **Edit query SQL** section.

> **Notice**
>
> - DataService Studio allows you to switch only from the codeless UI to the code editor.
> - After you switch from the codeless UI to the code editor, you cannot switch back to the codeless UI.

2.14.5.3. Create an API in the code editor

To meet the requirements of advanced data query, DataService Studio allows you to create an API by writing an SQL script in the code editor. DataService Studio supports table join queries, complex queries, and aggregate functions. This topic describes how to create an API in the code editor.

Prerequisites

Connections are configured on the **Data Source** page. For more information, see Configure data sources.

Create an API

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over and choose **API > Generate API**.You can also click a workflow, right-click **API**, and then choose **New > Generate API**.
4. In the **Generate API** dialog box, set the parameters as required.

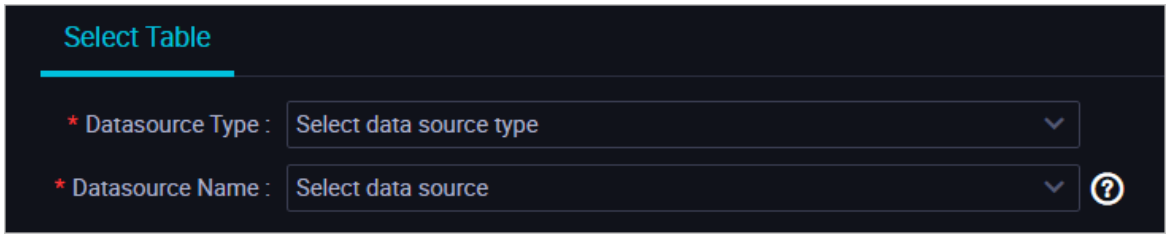
Parameter	Description
API mode	The mode for creating the API. Valid values: **Wizard Mode** and **Script Mode**. In this example, select **Script Mode**.
API Name	The name of the API. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the API, for example, */user*.
Call Mode	The mode for calling the API. Valid values: **Synchronous Call** and **Asynchronous Call**. ○ If you set this parameter to **Synchronous Mode**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ○ If you set this parameter to **Asynchronous Mode**, the API returns the RequestID parameter immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET** and **POST**.
Response Content Type	The return type of the API. Set the value to **JSON**.
Visible Range	The visibility of the API. Valid values: ○ **Work Space**: The API is visible to all members in the current workspace. ○ **Private**: The API is visible only to its owner and permissions on the API cannot be granted to other users. **Note** If you set the Visible Range parameter to Private, the API is visible only to you in the directory tree. It is hidden to other members of the workspace.

Parameter	Description
Label	The tag of the API. Select one or more tags from the drop-down list. For more information, see Manage tags. Note You can set at most five tags for an API.
Description	The description of the API, which can be up to 2,000 characters in length.
Target Folder	The directory for storing the API.

5. Click **OK**.

Configure the API

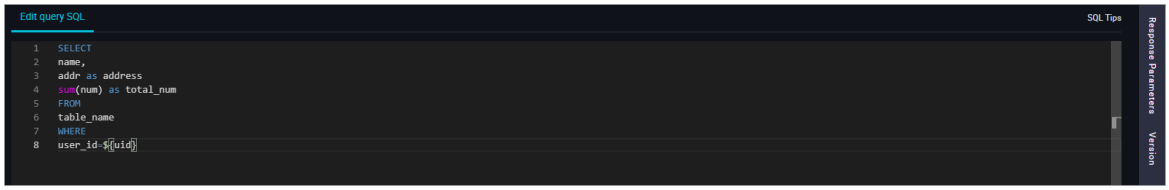
1. Double-click the API. On the configuration tab that appears, set the **Datasource Type** and **Datasource Name** parameters in the **Select Table** section.



Note You must configure a connection in Data Integration in advance. You can enter a keyword in the Table Name field to search for the desired table.

2. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.

3. In the **Edit query SQL** section, enter the SQL statement for querying data.

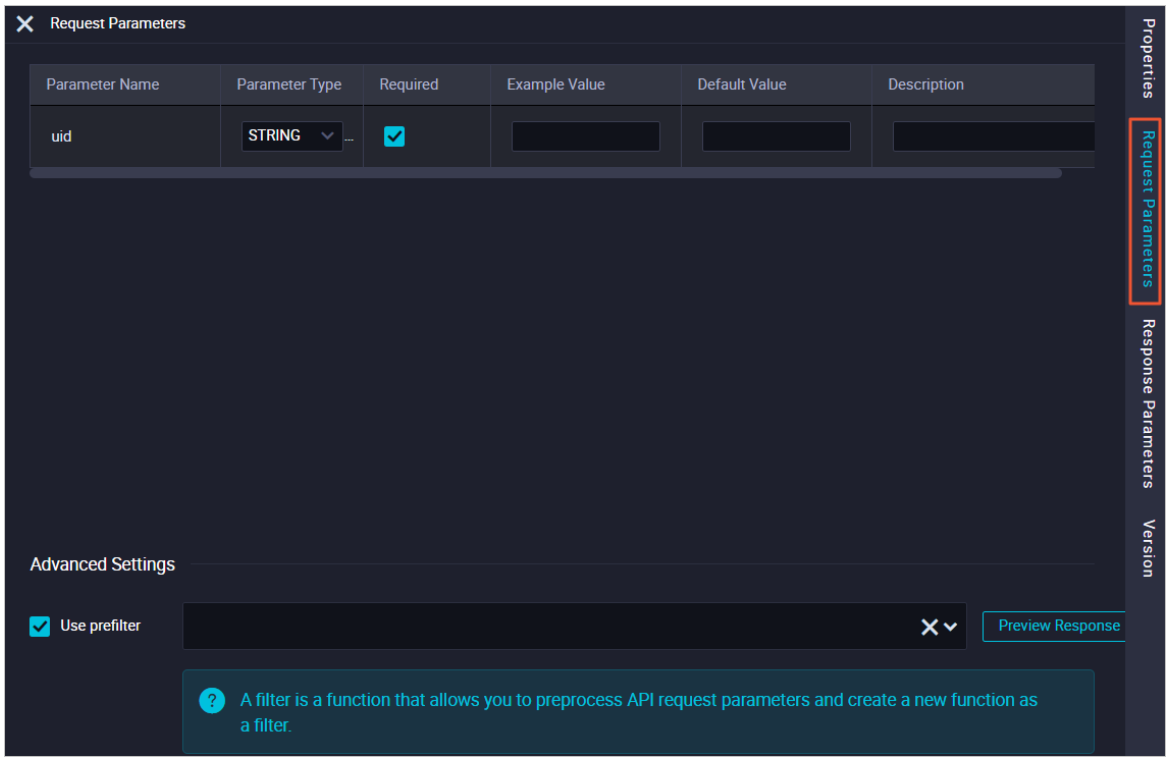


Note The SELECT clause specifies the parameters that the API returns. The WHERE clause specifies the request parameters of the API. You must use ${} to interpolate a request parameter.

Follow these rules when you enter the SQL statement:

- You can enter only one SQL statement in the script editor.
- Only the SELECT clause is supported. Other clauses such as INSERT, UPDATE, and DELETE are not supported.
- The clause `SELECT \*` is not supported. You must specify the columns to be queried.
- Single-table queries, table join queries, and nested queries under the same connection are supported.
- If the name of the column that the SELECT clause specifies has a table prefix, for example, t.name, you must create an alias for the corresponding response parameter, for example, t.name as name.
- If you use an aggregate function, such as min, max, sum, or count, you must create an alias for the corresponding response parameter, for example, `sum(num) as total\_num`.
- ${param} in the SQL statement and that in character strings are regarded as request parameters and replaced. If an escape character \ is placed before ${param}, ${param} is processed as a common string.
- ${param} cannot be enclosed in single quotation marks (' '). For example, `'${id}'` and `'abc ${xyz}123'` are not allowed. If necessary, you can use `concat('abc', ${xyz}, '123')` instead.
- Parameters cannot be configured as optional.
- ${param} is not allowed in comments. For example, --${id} is not allowed.

4. In the right-side navigation pane, click **Request Parameters**. In the Request Parameters pane, set the parameters as required.

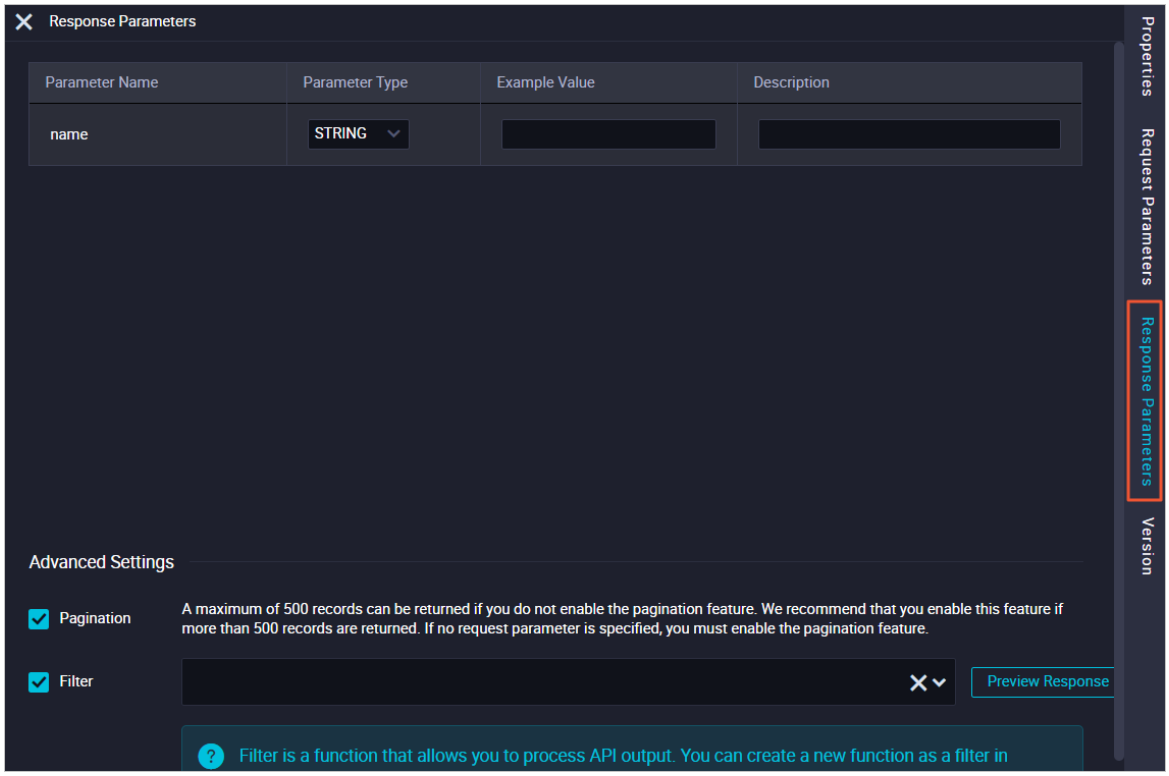


Parameter	Description
Parameter Name	The name of the request parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Parameter Type	The type of the request parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Required	Specifies whether the request parameter is required.
Example Value	The sample value of the request parameter.
Default Value	The default value of the request parameter.
Description	The description of the request parameter.

To preprocess the request parameters of the API, select Use prefilter in the Advanced Settings section. For more information, see Use prefilters.

> **Note**
>
> - To enhance the matching efficiency, set an indexed field as a request parameter.
> - To make it easier for API callers to know the details about the API, we recommend that you specify information such as the sample value, default value, and description for each parameter of the API.

5. **In the right-side navigation pane, click Response Parameters. In the Response Parameters pane, set the parameters as required.**



Parameter	Description
Parameter Name	The name of the response parameter. The name can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter and can be up to 64 characters in length.
Parameter Type	The type of the response parameter. Valid values: **STRING**, **INT**, **LONG**, **FLOAT**, **DOUBLE**, and **BOOLEAN**.
Example Value	The sample value of the response parameter.
Description	The description of the response parameter.

You can select **Pagination** and **Filter** in the **Advanced Settings** section.

Select Pagination based on your needs.

- If you do not select **Pagination**, the API returns a maximum of 2,000 records by default.
- If the API may return more than 2,000 records, we recommend that you select **Pagination**.

The following common parameters are available when Pagination is selected:

- Common request parameters
  - pageNum: the number of the page to return.
  - pageSize: the number of entries to return on each page.
- Common response parameters
  - pageNum: the page number of the returned page.
  - pageSize: the number of entries returned per page.
  - totalNum: the total number of returned entries.

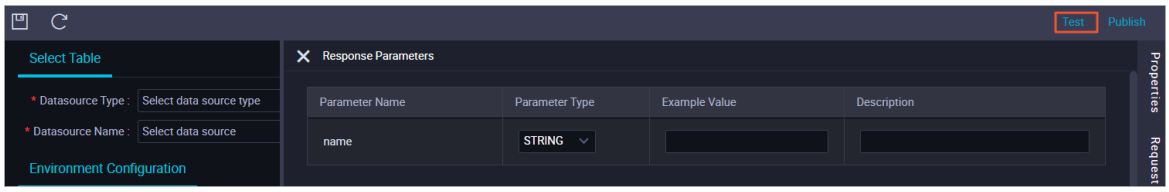
If you need to process the query results returned by the API, select **Filter**.

> **Note**
> - Field values are returned in the response as they are in the table.
> - Request parameters are optional for an API. If you do not specify any request parameters for an API, you must select **Pagination**.

6. Click Save icon in the toolbar.

Test the API

1. After you save the settings of the API, click **Test** in the upper-right corner.



2. In the **Test APIs** dialog box, click **Test** to send an API request.The request and response details appear on the right. If the API fails the test, check the error message, modify the API

settings accordingly, and test the API again.

You can select **Save the correct response example automatically** as required.

> **Note**
>
> - The system automatically generates sample failure responses and error codes when it tests an API. However, the system does not automatically generate sample success responses.
>
>   To allow the system to save the success test result as a sample success response, you must select **Save the correct response example automatically** before you perform the test. If the response contains sensitive data that must be de-identified, you can manually edit the response.
> - The sample success response is an important reference for API callers, and therefore must be configured.
> - The Call Latency value is the latency of the current API request, which is used to evaluate the API performance. If the latency is long, consider optimizing the database.

3. After the API is tested, close the **Test APIs** dialog box and click **Publish** in the upper-right corner of the configuration tab.

2.14.5.4. Use filters

2.14.5.4.1. Use prefilters

A prefilter is a function that is used to process request parameters of APIs. You can specify one or more prefilters to customize the request content for APIs. This topic describes the limits of prefilters, the built-in function template provided by the system, and how to create functions and use them as prefilters.

Context

Prefilters have the following limits:

- Only Python 3.0 functions can be used as prefilters.
- Prefilters support importing only the following modules: json, time, random, pickle, re, and math.
- The function name of a prefilter must be `def handler(event,context):` .

Function template

The system provides the following built-in function template:

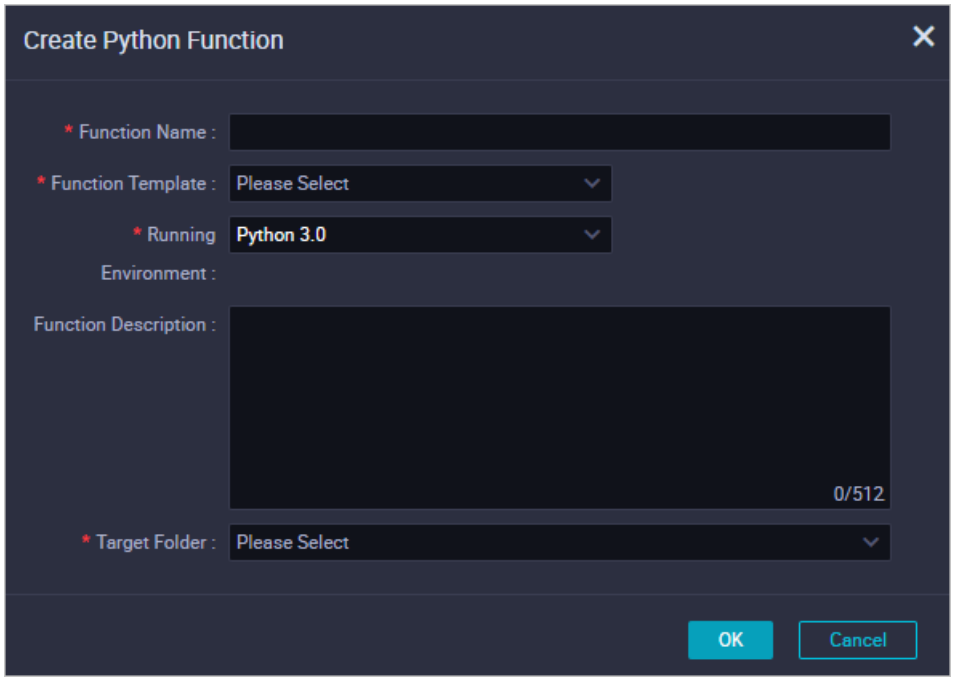
```
# -*- coding: utf-8 -*-


# event (str) : in filter it is the API result, in other cases, it is your param
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
# do not modify function name
import json
def handler(event,context):
# load str to json object
obj = json.loads(event) # Convert the string specified by the event parameter to a JSON object.
# add your code here
# end add
return obj
```

You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This parameter is n
ot in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is of the STRIN
G type.
```

Create a Python function

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over  and choose **Function > Create Python Function**. You can also click a workflow, right-click **Function**, and then choose **New > Create Python Function**.
4. In the **Create Python Function** dialog box, set the parameters as required.

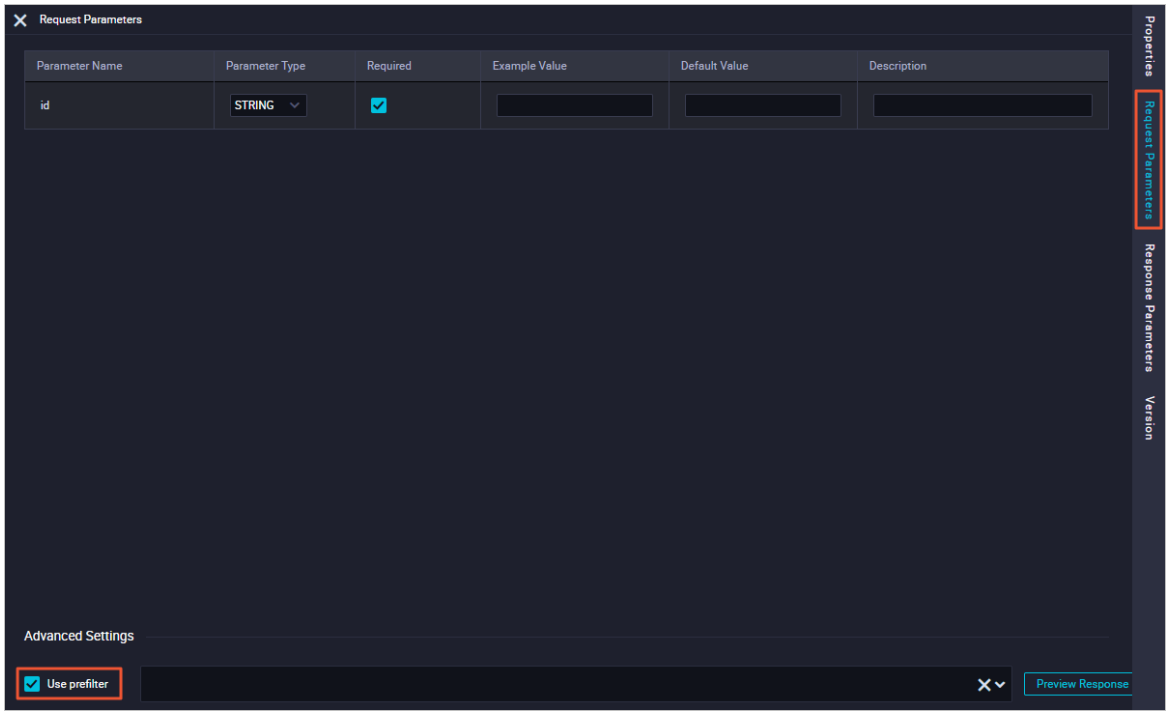


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function.
Target Folder	The directory for storing the function.

5. Click **OK**.

Use prefilters

1. On the **Service Development** tab, double-click the target API.
2. On the configuration tab that appears, click **Request Parameters** in the right-side navigation pane.
3. In the **Request Parameters** pane, select **Use prefilter** in the **Advanced Settings** section.



4. Select functions from the **Use prefilter** drop-down list.

   > **Note** A prefilter is a function that is used to process request parameters of APIs. You can create a function and use it as a prefilter.

5. Click **Preview Response** to view the processing results of the prefilters.

2.14.5.4.2. Use post filters

A post filter is a function that is used to process the results returned by APIs. You can specify one or more post filters to process the results returned by APIs. This topic describes the limits of post filters, the built-in function template provided by the system, and how to create functions and use them as post filters.

Context

Post filters have the following limits:

- Only Python 3.0 functions can be used as post filters.
- Post filters support importing only the following modules: json, time, random, pickle, re, and math.
- The function name of a post filter must be `def handler(event,context):` .

Function template

The system provides the following built-in function template:

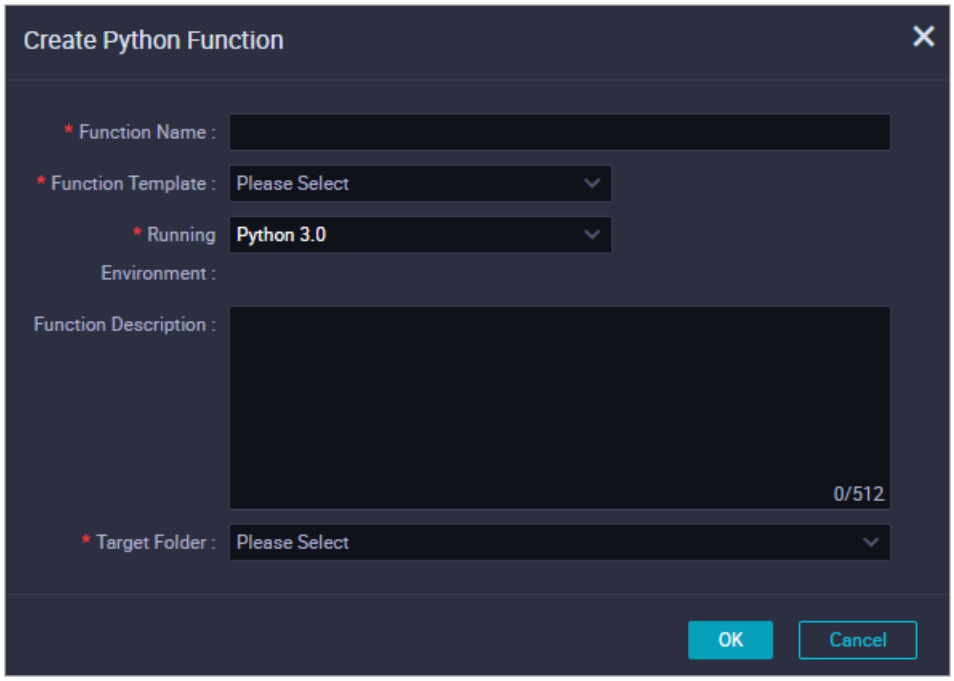
```
# -*- coding: utf-8 -*-


# event (str) : in filter it is the API result, in other cases, it is your param
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
# do not modify function name
import json
def handler(event,context):
# load str to json object
obj = json.loads(event) # Convert the string specified by the event parameter to a JSON object.
# add your code here
# end add
return obj
```

You can modify the function template to write your own function. You can modify the names of the input parameters as needed.

```
Parameter 1 [context]: the context of calling APIs. The value is of the STRING type. This parameter is n
ot in use and is left empty.
Parameter 2 [event]: the result data returned by APIs or the preceding filter. The value is of the STRIN
G type.
```

Create a Python function

1. Log on to the DataWorks console.
2. Click in the upper-left corner and choose **All Products > Data Service**.
3. On the **Service Development** tab, move the pointer over and choose **Function > Create Python Function**. You can also click a workflow, right-click **Function**, and then choose **New > Create Python Function**.
4. In the **Create Python Function** dialog box, set the parameters as required.

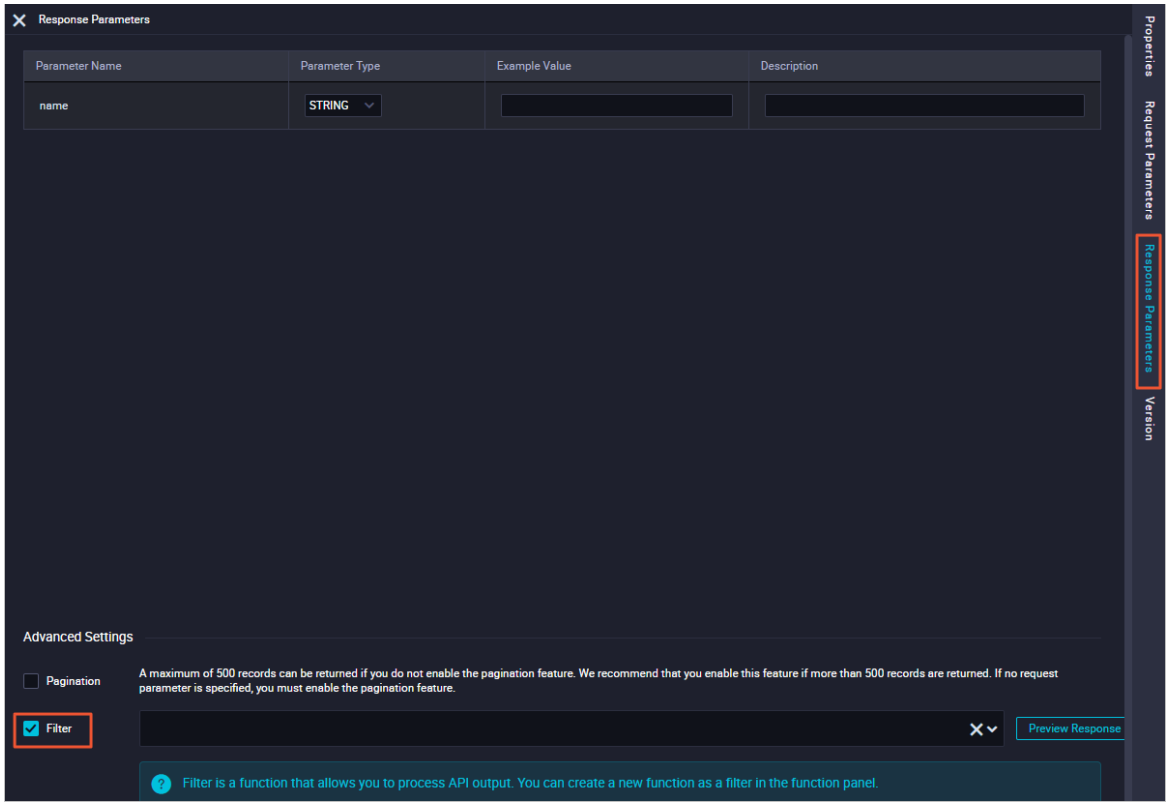


Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function.
Target Folder	The directory for storing the function.

5. Click **OK**.

Use post filters

1. On the **Service Development** tab, double-click the target API.
2. On the configuration tab that appears, click **Response Parameters** in the right-side navigation pane.
3. In the **Response Parameters** pane, select **Filter** in the **Advanced Settings** section.



4. Select functions from the **Filter** drop-down list.

   > **Note** A post filter is a function that is used to process the results returned by APIs. You can create a function and use it as a post filter.

5. Click **Preview Response** to view the processing results of the post filters.

2.14.6. Register APIs

This topic describes how to register APIs and manage and publish them to API Gateway together with APIs created based on data tables.

Currently, DataService Studio allows you to register only RESTful APIs. Supported request methods include GET, POST, PUT, and DELETE. Supported data types include forms, JSON data, and XML data.

Create a group

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products** > **DataService Studio**.
3. Go to the **Service Development** page, right-click **Service List**, and then select **New API Group**.
4. In the **Create API Group** dialog box that appears, enter values for **Group Name** and **Description**.
5. Click **OK**.

Configure basic API information

1. Right-click the new group and select **Register API**.
2. In the **Create API** dialog box that appears, set each parameter.

Configuration item	Description
API Name	The name must be 4 to 50 characters in length. It must start with a letter and can contain letters, digits, and underscores (_).
API Group	An API group is a collection of APIs for a specific feature or scenario. It is also the minimum API management unit of API Gateway. To create an API group, move the pointer over the **Create** icon and select **New API Group**.
API Path	API Path is the alias of Backend Service Path. APIs with different API paths can share the same backend service path and backend service host. Parameters defined in Backend Service Path must also be defined in brackets in API Path.
Protocol	Currently, HTTP and HTTPS are supported.
Request Method	You can select GET, POST, PUT, or DELETE as the request method. The parameters to be configured vary with the request method.
Return Type	Currently, JSON and XML return types are supported.
Description	The description of the API.

3. After configuring the basic API information, click **OK** to go to the API parameter configuration page.

Configure API parameters

On the API parameter configuration page, you need to define the backend service, request parameters, response content, and error codes.

Configuration item	Description
Define the back-end Service	• **Backend Service Host**: Enter the host of the API. The host must start with http:// or https://, and cannot contain the path. • **Backend Service Path**: Enter the path of the API. Place parameter names in brackets, for example, /user/[userid]. In the next step, parameters defined in Backend Service Path are automatically added to the request parameter list. • **Backend timeout**: Set the backend timeout period.

Configuration item	Description
Define Request Parameters	• **Parameter Type:** The available request parameter locations (Path, Header, Query, or Body) vary with the request method. • **Constant Parameters:** A constant parameter is fixed and is invisible to the caller. You do not have to specify the constant parameters when calling an API. Instead, the constant parameters and their values are automatically sent to the backend service. This is useful when you want to set a parameter to a fixed value and hide the parameter value from the caller. • **Request Body Definition:** This configuration item is available only when the request method is POST or PUT. You can enter the body description in Request Body Definition. It is equivalent to an example of the request body so that API callers can refer to the format of the request body. The content type of the request body can be JSON or XML. **Note** If a parameter has been defined in both the request body and the request parameter list, the parameter value in the request body takes priority.
Define Response Content	You can enter a successful response example or an error response example for API callers to refer to when writing the return parse code.
Error Codes	Enter the common errors and solutions in API calling. This enables API callers to troubleshoot and solve these errors. **Note** To ensure that the API is easily used by the callers, provide complete API parameter information if possible, especially the parameter sample values, default values, and sample responses.

Test the API

After you have configured and saved the API parameters, click **Test** in the upper-right corner to go to the API test page.

Set the parameters and click **Test** to send an API request. The request and response details are displayed on the right. If the test fails, check carefully the error message, make modifications accordingly, and test the API again.

Pay attention to the settings of the successful response example in the configuration process. When testing an API, the system automatically generates error response examples and error codes. However, successful response examples are not automatically generated. To enable the system to save the test result as the successful response example, you need to select **Save as Successful Response Example** before performing the test. If the response contains sensitive data that must be masked, you can manually edit the response.

> Note
>
> - The successful response example is an important reference for API callers, and therefore must be configured.
> - The Call Latency value is the latency of the current API request, which is used to evaluate the API performance. If the latency is long, consider optimizing the database.

After the API test is passed, return to the **Service Development** page, and click **Publish** to generate an API.

2.14.7. Test APIs

This topic describes how to test APIs.

When creating and registering an API, you can test the API. The system also provides an independent API test feature, which allows you to test APIs online.

1. Go to the **DataService Studio** page and click **Service Management** in the upper-right corner.
2. Click **Test API** in the left-side navigation pane.
3. Select the API to be tested, set the parameters, and then click **Test**.

> Note
>
> - The API Test page provides only online API testing. You cannot update or save the successful response example for an API on this page. To update the successful response example for an API, click the API name in the API list to enter the API edit mode. Then, update the successful response example for the API in the API testing step.
> - You must test an API before publishing it.

2.14.8. Publish APIs

API Gateway provides API lifecycle management services, including API publishing, management, maintenance, and monetization. It helps you easily and quickly aggregate microservices, separate the frontend from the backend, and integrate systems at low costs and low risks, making features and data available to partners and developers.

API Gateway provides permission management, traffic control, access control, and metering services. The services make it easy for you to create, monitor, and secure APIs. Therefore, we recommend that you publish the APIs that have been created and registered in DataService Studio to API Gateway. DataService Studio and API Gateway are interconnected, which allows you to publish APIs to API Gateway easily.

Publish APIs to API Gateway

Before publishing an API, you must activate API Gateway and register and test the API.

After the API passes the test, click **Publish** in the upper-right corner to publish the API to API Gateway.

The system automatically registers the API with API Gateway during the publish process. The system also creates a group in API Gateway with the same name as the API group to which the API belongs in DataService Studio and publishes the API in this group. After the API is published, you can access the API Gateway console to view API details or configure bandwidth throttling, access control, and other features.

If you generate an API to be called by your own application, you need to create an application in API Gateway, authorize the application to use the API, and enable the application to call the API by using AppKey and AppSecret. For more information, see *API Gateway documentation* . API Gateway also provides SDKs for mainstream programming languages to help you quickly integrate the API into your own application.

2.14.9. Call APIs

This topic describes how to call an API after this API is published on API Gateway.

API Gateway provides the SDKs for authorizing and calling APIs. You can authorize yourself, your associates, or third parties to use APIs. If you want to call an API, perform the following operations.

Three conditions for calling an API

The following three conditions must be met to call an API:

- API: The API that you call is clearly defined by API parameters.
- App: The app that you use to call the API has a key pair that uniquely identifies you. The key pair consists of the AppKey and AppSecret.
- Relationship between the API and app: If you want to call an API by using an app, the app must have the permission to call this API. This permission is granted through authorization.

Procedure

1. Obtain the API documentation.

   The method of obtaining the API documentation varies depending on how you obtain the API. You can obtain the API by purchasing it from the marketplace, or you are authorized to use the API for free.

2. Create an app.

   The app identifies you when you call the API. Each app has a key pair: AppKey and AppSecret, which are equivalent to an account and the corresponding password.

3. Obtain the permission to call the API.

   Authorization means granting the app the permission to call an API. Your app must be authorized to call the API. The authorization method varies depending on how you obtain the API.

4. Call the API.

   You can edit an HTTP or HTTPS request to call the API. Before calling the API, you can use examples of calling APIs in multiple languages in the API Gateway console to test the call.

You can call APIs after you passed the quick authentication or encrypted signature authentication.

Click **API call**. The AppCode and AppKey that are synchronized from API Gateway are available on the page that appears. You can perform **copy** and **reset** operations on them. For more information about API Gateway, see *API Gateway documentation* .

2.14.10. Use workflows

The workflow feature, also called service orchestration, provides a composite API service. This topic describes the benefits of workflows and how to use workflows.

In DataService Studio, a workflow is represented as a directed acyclic graph (DAG). By dragging and dropping nodes to a DAG, you can arrange APIs and functions in a serial, parallel, or branch structure based on the business logic.

When you run a workflow to call APIs, DataWorks runs the nodes in the workflow in sequence, passes parameters among the nodes, and automatically changes the status of each node. The workflow feature simplifies the process of calling multiple APIs or functions and reduces the cost of development and maintenance. This allows you to focus on business development.

Benefits

- Reduced cost of combining multiple APIs

  By dragging and dropping nodes to a DAG, you can arrange APIs and functions in a serial, parallel, or branch structure without writing code. This reduces the cost of developing APIs.

- Higher performance in calling APIs and functions

  A workflow allows you to call multiple APIs and functions in a container. Compared with writing code to call APIs and functions, the workflow feature reduces the latency of calling APIs and functions.

- Serverless architecture

  The workflow feature adopts a serverless architecture. A serverless architecture supports automatic resource scaling based on business needs. You can focus on the business logic, without worrying about the runtime environment.

Obtain values of request and response parameters

DataService Studio uses JSONPath to obtain parameter values. JSONPath is a query language that allows you to extract data from JSON files.

For example, three nodes are run in the following order: A, B, and then C. Node C needs to use the response parameters of nodes A and B.

- Response parameter of node A: {"namea":"valuea"}

  Expression for obtaining the value of the response parameter of node A: ${A.namea}

- Response parameter of node B: {"nameb":"valueb"}

  Expression for obtaining the value of the response parameter of node B: $.nameb or ${B.nameb}

The built-in start node provides request parameters for the whole workflow. Assume that a request parameter of a workflow is {"namewf":"valuewf"}. All nodes of the workflow can obtain the value of the request parameter by using the ${START.namewf} expression.

Set parameters

Request parameters:

- If you do not specify a value for a request parameter of a node, DataService Studio obtains the value of the same parameter in the first layer of the JSON string returned by the parent node, and assigns the value to the request parameter. If no value is specified for a request parameter of the first node, DataService Studio obtains the value of the same parameter in the request parameters of the workflow.
- If you specify a value for a request parameter, DataService Studio uses the value that you set.
- If you need to use the value of the specified parameter returned by a specified ancestor node, obtain the value by using a JSONPath expression.

Common JSONPath expressions for obtaining parameter values:

- $.: obtains response parameters of the parent node.
- $.param: obtains the value of the param parameter returned by the parent node. To allow you to obtain response parameters of any ancestor nodes, DataService Studio enhances JSONPath expressions.
- ${NODEID1}: obtains response parameters of the node whose ID is NODEID1.
- ${START}: obtains request parameters of the workflow, which are response parameters of the start node.
- ${NODEID1.param}: obtains the value of the param parameter returned by the node whose ID is NODEID1.

JSONPath expressions for setting response parameters of a node:

- $.: sets response parameters of the current node.
- $.param: sets the param parameter to be returned by the current node.
- ${NODEID1.param}: sets the param parameter returned by the node whose ID is NODEID1.

Example

Add a connection before you create and use a workflow. In this example, a MySQL connection is used.

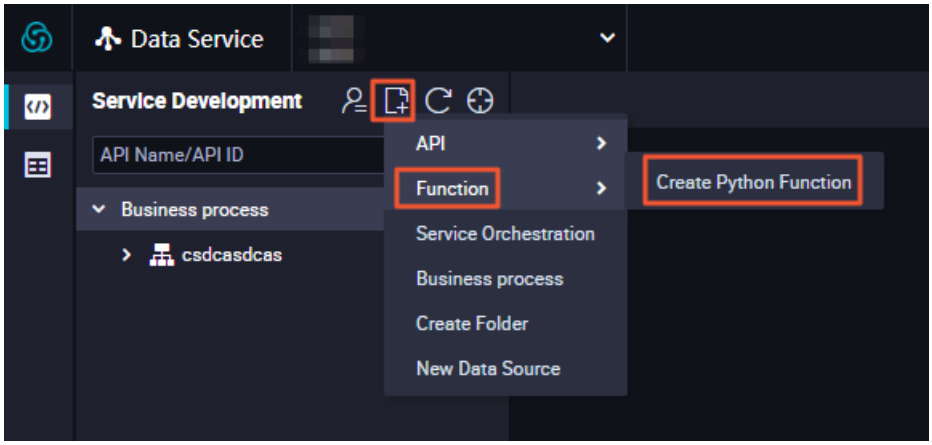
1. Register an API.

   In this example, create an API by using the registration method. For more information, see Register an API.

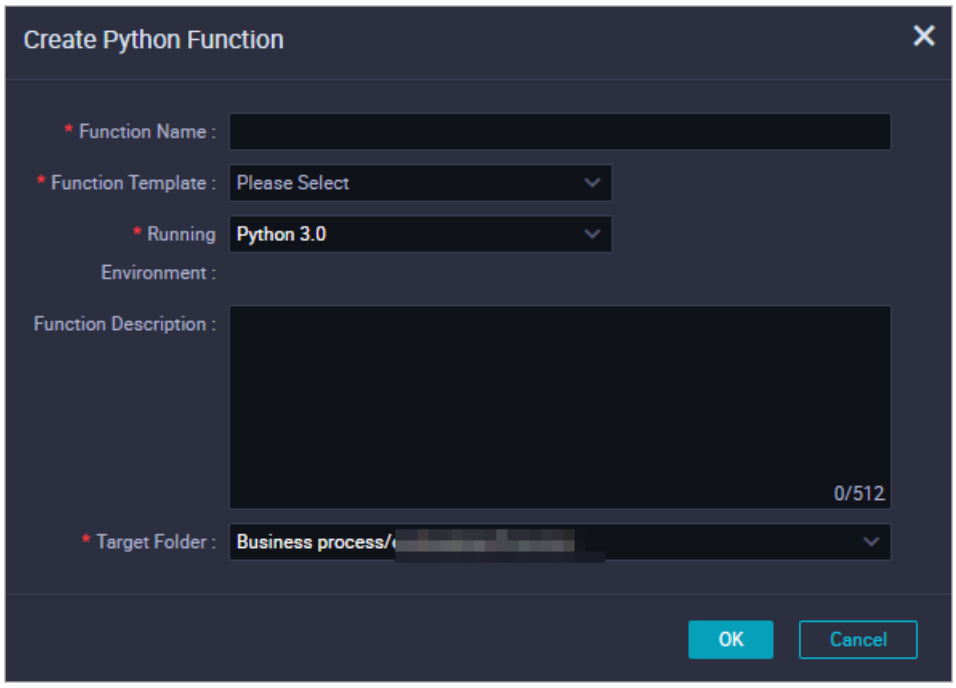
2. Register a function. For more information, see Create a function.

   In this example, create a Python function as a branch node to process the result data of the parent node.

i. Go to the **Service Development** tab, move the pointer over and choose **Function** > **Create Python Function**.



ii. In the **Create Python Function** dialog box, set the parameters as required.



Parameter	Description
Function Name	The name of the function to create, which can be up to 256 characters in length.
Function Template	The template used to create the function. Set the value to Python3 Standard v1.
Running Environment	The runtime environment of the function. Set the value to Python 3.0.
Function Description	The description of the function. The description can be up to 512 characters in length.
Target Folder	The directory for storing the function.

iii. Click **OK**.

iv. On the configuration tab of the function, configure the function.

a. In the **Edit Code** section, enter the function code.

```
# -*- coding: utf-8 -*-
# event (str) : in filter it is the API result, in other cases, it is your param
# context : some environment information, temporarily useless
# import module limit: json,time,random,pickle,re,math
import json
def handler(event,context):
    # load str to json object
    obj = json.loads(event)
    # add your code here
    # end add
    return obj
```

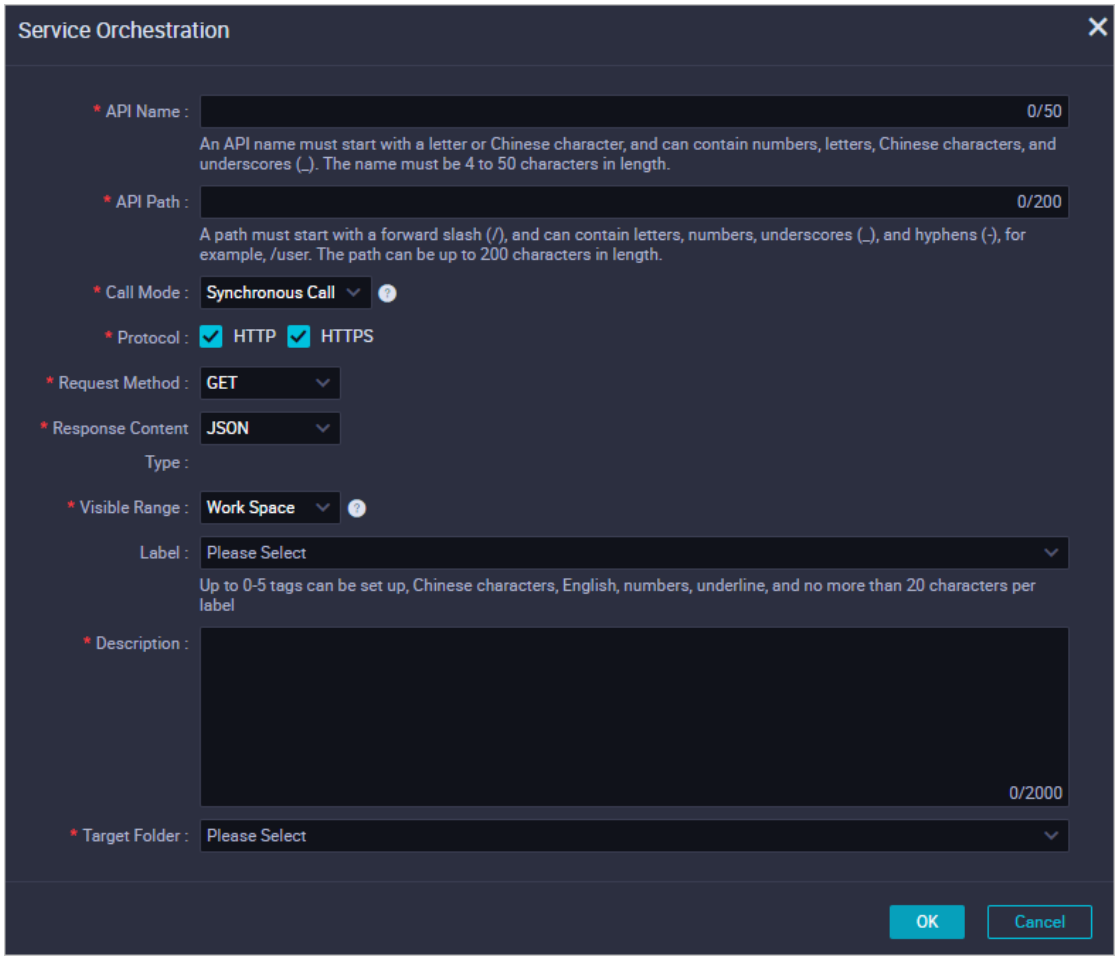
b. In the **Environment Configuration** section, set the **Memory** and **Function Timeout** parameters.

v. Click Save icon in the toolbar.

3. Create a workflow.

i. On the **Service Development** tab, move the pointer over and select **Service Orchestration**.

ii. In the **Service Orchestration** dialog box, set the parameters as required.



Parameter	Description
API Name	The name of the workflow. The name must be 4 to 50 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.
API Path	The path for storing the workflow, for example, */user*.
Call Mode	The mode for calling the API to be arranged in the workflow. Valid values: **Synchronous Call** and **Asynchronous Call**. ■ If you set this parameter to **Synchronous Mode**, the API returns results immediately after it is called. The synchronous mode is most commonly used. ■ If you set this parameter to **Asynchronous Mode**, the API returns the RequestID parameter immediately after it is called. The API caller can then obtain the call result from a message queue based on the request ID.
Protocol	The protocol used by the API. Valid values: **HTTP** and **HTTPS**.
Request Method	The request method used by the API. Valid values: **GET** and **POST**.
Response Content Type	The return type of the API. Set the value to **JSON**.

Parameter	Description
Visible Range	The visibility of the workflow. Valid values: ■ **Work Space**: The workflow is visible to all members in the current workspace. ■ **Private**: The workflow is visible only to its owner and permissions on the workflow cannot be granted to other users. Note If you set the Visible Range parameter to Private, the workflow is visible only to you in the directory tree. It is hidden to other members of the workspace.
Label	The tag of the workflow. Select one or more tags from the drop-down list. For more information, see Manage tags. Note You can set at most five tags for a workflow.
Description	The description of the workflow, which can be up to 2,000 characters in length.
Target Folder	The directory for storing the workflow.

iii. Click **OK**.

4. Configure the workflow.

i. On the configuration tab of the workflow, drag and drop nodes and connect them as required.

ii. Double-click the **API1** node to edit the node. Select the API that you registered earlier as the API to be called in the node.

Select **set output results** and enter `{"user_id":"$.data[0].id"}` .

Use JSONPath expressions to set response parameters. The syntax for obtaining the value of a parameter is ${NodeA.namea}, which is the same as that for setting request parameters. `{" user_id":"$.data[0].id"}` assigns the value of the id parameter of the first element in the data array to the user_id parameter. Then, the API1 node returns {"user_id":"value"} in JSON format.

iii. Double-click the **PYTHON1** node to edit the node. Select the function that you created earlier as the function to be called in the node.

iv. Double-click the **SWITCH2** node to edit the node. In the right-side pane that appears, click **Set branch conditions**. In the Set branch conditions dialog box, enter conditional expressions based on the response parameter of the parent node. For example, you can enter expressions in the `${Node ID. Parameter}>1` or `$. Parameter>1` format. Conditional expressions support the following operators: ==, ! =, >=, >, <=, <, &&, !, (), +, -, *, /, and %.

In this example, the user_id parameter is the response parameter of the API1 node and is used as the request parameter of the SWITCH2 node.

```
Branch node 1: $.user_id != 1, indicating that the branch node 1 is run if the value of the user
_id parameter is not 1.
Branch node 2: $.user_id == 1, indicating that the branch node 2 is run if the value of the user
_id parameter is 1.
```

v. Double-click the end node and then click the **Response Parameters** tab on the right side to set response parameters.

5. Click **Test** in the upper-right corner.
6. In the **Test APIs** dialog box, set the parameters as required and click **OK**.

   You can view the test result after the workflow is tested.

2.14.11. Manage versions

This topic describes how to manage versions of APIs, workflows, and functions in Data Service.

You can view and compare historical versions of APIs, workflows, and functions. Data Service generates a version record each time an API, a workflow, or a function is published.

View historical versions

1. Log on to the DataWorks console. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > DataService Studio**.
2. On the **Service Development** page that appears, double-click the name of the target API in the API list.

   > **Note** You can also click Function or Table in the left-side navigation pane and double-click the name of a workflow or function to manage its versions.

3. In the right-side navigation pane, click **Version**. In the **Version** dialog box that appears, view historical versions of the API.

Parameter	Description
API ID	The ID of the API. Each API ID is unique.
Version	The version of the API. A version is generated each time the API is published. V1 indicates version 1 and V2 indicates version 2. The version number is incremented by 1 each time.

Parameter	Description
The Author	The user who published the version.
Submitted Date	The time when the version was published. The time is accurate to second.
Status	The status of the version. Value values: ○ **Release**: indicates that the version of the API is the latest version. ○ **Off-Line**: indicates that the version of the API is a historical version.
Actions	The operations that you can perform on the version. The **Actions** column is only available for API and workflow versions. You can click **API Details** of a version to go to the details page of the version. The **Actions** column is unavailable for function versions.

Compare historical versions

In the **Version** dialog box, select two versions to compare, and click **Contrast**. In the **History Version Contrast** dialog box that appears, compare the code and parameters of the two versions.

2.14.12. FAQ

This topic provides answers to commonly asked questions about DataService Studio.

- Q: Do I need to activate the API Gateway service?

  A: API Gateway provides you with high-performance and highly available API hosting services. If you need to make your APIs available to others, activate the API Gateway service first.

- Q: Where can I add and change connections?

  A: After you log on to the DataWorks console, click the DataWorks icon in the upper-left corner and choose **All Products** > **Data Integration** to go to the **Data Integration** page. In the left-side navigation pane, choose **Sync Resources** > **Connections**. On the page that appears, perform the relevant configuration. DataService Studio automatically reads data from the connections that you have configured.

- Q: What is the role of an API group in DataService Studio? What is the relationship between an API group in DataService Studio and an API group in API Gateway?

  A: An API group is a set of APIs specific to a feature or scenario. It is the smallest organization unit in DataService Studio, which is similar to an API group in API Gateway. An API group in DataService Studio is equivalent to an API group in API Gateway. After you publish an API from DataService Studio to API Gateway, API Gateway automatically creates an API group with the same name.

- Q: How can I configure an API group appropriately?

A: Typically, an API group includes APIs that provide similar features or resolve a specific issue. For example, a weather API group can include APIs that are used to check the weather by city and by longitude and latitude.

- Q: How many API groups can I create?

  A: An Alibaba Cloud account can create up to 100 API groups.

- Q: When do I need to enable the pagination feature for an API call so that its return results can be displayed on multiple pages?

  A: By default, an API call returns a maximum of 2,000 records. If an API call may return more than 2,000 records, enable the pagination feature. If you do not specify any request parameters, the API call usually returns a large number of records and the pagination feature is automatically enabled.

- Q: Do APIs created by DataService Studio support POST requests?

  A: APIs created by DataService Studio support GET and POST requests.

- Q: Do APIs created by DataService Studio support the HTTPS protocol?

  A: APIs created by DataService Studio support both HTTP and HTTPS protocols.

2.15. Stream Studio

2.15.1. Overview

Built on Alibaba Cloud Realtime Compute, which is based on Apache Flink, Stream Studio allows you to develop real-time computing nodes in directed acyclic graph (DAG) mode or SQL mode. You can switch between the two modes to edit the code or drag and drop components and configure them in a visual way.

As an ideal platform for developing real-time computing nodes, Stream Studio has the following features:

- Supports developing nodes in DAG mode. You can perform drag and drop components to configure real-time computing nodes.
- Supports developing nodes in SQL mode. You can edit the code to configure real-time computing nodes.
- Supports switching between the DAG mode and the SQL mode for you to easily check SQL operators.
- Supports using Function Studio to create and publish user-defined functions (UDFs) online in exclusive mode.
- Supports smart diagnosis for real-time computing nodes to facilitate online troubleshooting.

2.15.2. Bind a Realtime Compute project

1. Log on to the DataWorks console.
2. Click the **Project Manage** icon in the upper-right corner. The **Project Management** page appears.
3. On the **Project Management** page, click **Add Compute Engine** in the **Compute Engine** section, and select **Add engine service**.
4. Enter the name of the Blink engine to be added and click **Bind**.

2.15.3. Create a real-time computing node

This topic describes how to create a real-time computing node and develop data in Stream Studio.

Prerequisites

A workflow is created. You can create real-time computing nodes and develop data under an existing workflow.

Procedure

After a workflow is created, you can create real-time computing nodes under the workflow. By default, data is developed for a real-time computing node in directed acyclic graph (DAG) mode.

1. Right-click the workflow that you have created and select **Create task**.
2. In the **Create Node** dialog box that appears, set the parameters and click **Submit**.
3. Develop data in DAG mode on the **Components** page.

   The Components page includes the following four sections:

   - Component list section: In this section, you can view the list of available components. You can click **Components** on the left-side navigation submenu to go to the Components page and view the list.
   - DAG section: In this section, you can drag and drop components to the DAG and connect them. To configure the dependency between two components, click and hold the highlighted dot at the bottom of a component and move the pointer to link this component with a descendant component. A DAG corresponds to a real-time computing node.
   - Parameter configuration section: Double-click a component in the DAG. Then, you can set the related parameters in this section.
   - Toolbar section: In this section, you can click the icons to perform the save, submit, steal lock, pre-compilation, test, stop, reload, and format operations respectively.

   When you configure the DAG, you can right-click a component and select an operation from the menu that appears to perform it on the selected component. Available operations include **Rename**, **View schema**, **Delete node**, **View error message**, **New component group**, and **Copy**.

2.15.4. Get started with Stream Studio

This topic uses an example to describe how to use Stream Studio to develop and manage a real-time computing node. Stream Studio allows you to create, configure, publish, run, stop, and unpublish a real-time computing node.

Prerequisites

A Realtime Compute project is bound to the current DataWorks workspace.

Context

- Data store: a Datahub table with a created topic, which contains the `m_name, id, m_type, and t`

`ag` fields.

> **Note** The Datahub topic must be created in advance.

- Data processing: splits the tag field by using the `semicolon (;)` as the delimiter to the color, mode, and weight fields.
- Output data: writes the `id, m_type, and weight` fields to a Log Service table.

> **Note** The Log Service project and Logstore must be created in advance.

Procedure

1. Log on to the DataWorks console. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Stream Studio**.
2. Create a workflow.
   i. On the Stream Studio homepage, click **New business process**.

      You can also move the pointer over the **Create** icon and click **Business process**.

   ii. In the **Create business process** dialog box that appears, set the **Business Name** and **Description** parameters.

   iii. Click **New**.
3. Create a real-time computing node.
   i. Right-click the workflow that you have created and select **Create task**.

   ii. In the **Create node** dialog box that appears, set the relevant parameters.

   iii. Click **Submit**. The **Components** page appears.

   iv. On the left-side navigation submenu, click **Resource Reference** and select **PUBLIC_COMMON**.

   > **Note** If this option is not selected, the message shown when you use the **FixedFieldsSplit** component.
   >
   > You can also go to the Resource Reference page to configure the reference resources after this message appears.
4. Configure the created real-time computing node.

   On the left-side navigation submenu, click **Components**. The **Components** page appears.

   i. Configure the data store of the node on the **Components** page.

      a. Drag and drop the Datahub component in the Data Source section to the directed acyclic graph (DAG).

      b. Click the Datahub component and set the parameters as needed on the **Parameter configuration** tab that appears.

Parameter	Description

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name. If this parameter is not specified, the real table name is used.
table schema	The custom fields and attribute fields to be returned. Click **Custom**. In the **Select field** dialog box that appears, click **+ Add** and enter the name and type of the output field. Then, click **OK**.
endPoint	The endpoint used to access Datahub. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to read data from Datahub. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to read data from Datahub. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Datahub project from which data is to be read. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Datahub topic from which data is to be read. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
startTime	The beginning of the time range when data is read. It corresponds to the startTime parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for reading data from Datahub. It corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement. Default value: *20*.
retryIntervalMs	The retry interval at which data is read. It corresponds to the retryIntervalMs parameter in the WITH clause of the CREATE TABLE statement. Unit: milliseconds. Default value: *1,000*.
batchReadSize	The number of data records that are read at a time. It corresponds to the batchReadSize parameter in the WITH clause of the CREATE TABLE statement. Default value: *10*.
lengthCheck	The rule for checking the number of fields parsed from a row of data. It corresponds to the lengthCheck parameter in the WITH clause of the CREATE TABLE statement. Default value: *NONE*.

Parameter	Description
columnErrorDebug	Specifies whether to enable debugging. It corresponds to the columnErrorDebug parameter in the WITH clause of the CREATE TABLE statement. If you turn on this switch, logs about parsing errors are returned. You can view the node details Operation Center.

ii. Configure the data operator.

a. Drag and drop the **FixedFieldsSplit** component to the DAG to split the tag field.

b. Click and hold the highlighted dot at the bottom of the **Datahub** component and move the pointer to link this component with the **FixedFieldsSplit** component.

c. Click the **FixedFieldsSplit** component and set the field to tag and the column separator to semicolon (;) on the Parameter configuration tab that appears.

d. Click **Custom** for the Add column parameter. In the **Select field** dialog box that appears, click + Add and enter the name and type of the output field. Then, click **OK**.

e. Drag and drop the **Select** component to the DAG. Click and hold the highlighted dot at the bottom of the **FixedFieldsSplit** component and move the pointer to link this component with the **Select** component.

f. Click the **Select** component and click **0 Field has been selected** on the **Parameter configuration** tab that appears.

g. In the dialog box that appears, select the fields to be returned and click **OK**.

iii. Configure the result table.

This example uses the LogService component as the destination.

a. Drag and drop the **LogService** component to the DAG. Click and hold the highlighted dot at the bottom of the **Select** component and move the pointer to link this component with the **LogService** component.

b. Click the **LogService** component and set the parameters as needed on the **Parameter configuration** tab that appears.

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name. If this parameter is not specified, the real table name is used.
Output Field	The fields to be returned. Click **0 Field has been selected** for the Output Field parameter. In the dialog box that appears, select the fields to be returned and click **OK**.
endPoint	The endpoint used to access Log Service. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Log Service project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Log Service topic to which data is to be written. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
source	The name of the Log Service table to which data is to be written. It corresponds to the source parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Log Service. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access Log Service. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
mode	The mode of data writing. It corresponds to the mode parameter in the WITH clause of the CREATE TABLE statement. Default value: *random*.
logStore	The name of the Logstore in the Log Service project to which the data is to be written. It corresponds to the logStore parameter in the WITH clause of the CREATE TABLE statement.

iv. Switch between the DAG mode and SQL mode.

Stream Studio allows you to configure a real-time computing node in both DAG mode and SQL mode. You can switch between these two modes.

By default, you configure a node in DAG mode. You can click **Switch to SQL mode** in the upper-right corner to switch to the SQL mode.

In SQL mode, you can click **Switch to DAG mode** in the upper-right corner to switch back to the DAG mode.

v. Configure the execution plan.

a. Click **Execution Plan** on the right-side navigation submenu to generate an execution plan.

b. Click **Save execution plan**.

5. Publish the real-time computing node.

You can publish the real-time computing node that you have configured. Click **Save** and then click **Submit** to publish the node.

i. Click **Save** and then click **Submit**. If you have not saved the node, a message appears, indicating that you must save it.

ii. In the **Submit New version** dialog box that appears, enter the remarks for the node and click **OK**.

iii. After you publish the node, you can go to the **OAM** page to view the node status and manage the node.

6. Perform O&M on the real-time computing node.

Click **OAM** in the upper-right corner to perform O&M on the real-time computing node.

i. Start the real-time computing node.

Find the real-time computing node that you have created in the node list and click **Start** to start the node.

You can set a custom start time for the real-time computing node based on your business requirement.

After starting the real-time computing node, you can click the node name to view its running status. If the real-time computing node is started properly, it enters the **Run** state.

ii. Stop and unpublish the real-time computing node.

a. Click **Stop** to stop the real-time computing node.

b. After the real-time computing node is stopped, click **Offline** to unpublish it.

Result

Now you have created, configured, published, run, stopped, and unpublished a real-time streaming node.

2.15.5. Configure components

2.15.5.1. Source tables

2.15.5.1.1. Datahub

Datahub is a real-time data distribution platform that is designed to process streaming data. It provides a channel for the Apsara Stack DTplus platform to process big data.

Realtime Compute typically uses Datahub to store source and result tables for streaming data processing. Data Transmission Services (DTS) and the Internet of Things (IoT) also use Datahub to access big data platforms. Datahub stores streaming data that can be used as input data for Realtime Compute.

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
table schema	The custom fields and attribute fields to be read from Datahub.	None
endPoint	The consumer endpoint.	None
accessId	The AccessKey ID used to read data from Datahub.	None
accessKey	The AccessKey secret used to read data from Datahub.	None
project	The name of the Datahub project from which data is to be read.	None
topic	The name of the Datahub topic from which data is to be read.	None
startTime	The beginning of the time range when data is read.	The format is `yyyy-MM-dd hh:mm:ss`.
maxRetryTimes	The maximum number of retries for reading data from Datahub.	None
retryIntervalMs	The retry interval at which data is read. Unit: milliseconds.	None
batchReadSize	The number of data records that are read at a time.	None

Parameter	Description	Remarks
lengthCheck	The rule for checking the number of fields parsed from a row of data.	Valid values: *SKIP*, *EXCEPTION*, and *PAD*. Default value: SKIP. • *SKIP*: skips a data record when the number of fields in the data record is not the specified one. • *EXCEPTION*: throws an exception when the number of fields in the data record is not the specified one. • *PAD*: pads fields in sequence. Pad a field with null when the field does not exist.
columnErrorDebug	Specifies whether to enable debugging. If you turn on this switch, logs about parsing errors are returned.	None
BLOB	Specifies whether the type of data read from Datahub is BLOB.	None
Data Quality	Specifies whether to open the Data Quality page to view related monitoring nodes.	None

Field type mapping

The following table lists the mapping between Datahub and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Datahub data type	Realtime Compute data type
BIGINT	BIGINT
DOUBLE	DOUBLE
TIMESTAMP	BIGINT
BOOLEAN	BOOLEAN
DECIMAL	DECIMAL

Attribute fields

You can obtain the attribute field indicating the system time at which each data record is written to Datahub.

Field	Description
System Time	The system time at which each data record is written to Datahub.

2.15.5.1.2. Log Service

As an all-in-one real-time data logging service, Log Service allows you to quickly finish tasks such as data ingestion, consumption, delivery, query, and analysis without any extra development work. This can help you improve O&M and operational efficiency, and build up the capability to process large amounts of logs in the data technology era.

Log Service stores streaming data that can be used as input data for Realtime Compute.

The data format of Log Service is consistent with JSON. Example:

```
{
  "a": 1000,
  "b": 1234,
  "c": "li"
}
```

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
table schema	The custom fields and attribute fields to be read from Log Service.	None
endPoint	The consumer endpoint.	Log Service endpoints
accessId	The AccessKey ID used to read data from Log Service.	None
accessKey	The AccessKey secret used to read data from Log Service.	None
project	The name of the Log Service project from which data is to be read.	None
logStore	The name of the Logstore under the Log Service project.	None

Parameter	Description	Remarks
consumerGroup	The name of the consumer group.	You can specify a custom consumer group name. The format of the name is not fixed.
startTime	The beginning of the time range when the log data is consumed.	None
heartBeatIntervalMills	Optional. The heartbeat interval at which the client sends heartbeat messages. Unit: milliseconds.	None
maxRetryTimes	The maximum number of retries for reading data from Log Service.	None
columnErrorDebug	Specifies whether to enable debugging. If you turn on this switch, logs about parsing errors are returned.	None

Field type mapping

The following table lists the mapping between Log Service and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Log Service data type	Realtime Compute data type
STRING	VARCHAR

Attribute fields

Currently, Log Service supports the following three attribute fields by default. You can also specify other custom fields.

Field	Description
`__source__`	Specifies a log source.
`__topic__`	Specifies a log topic.
`__timestamp__`	Specifies the time when a logged event occurs.

Note

- Currently, Log Service does not support the MAP type.
- We recommend that you define the fields in the same order as the fields in the preceding table. Unordered fields are also supported.
- If the input data is in JSON format, define the delimiter and use the built-in function JSON_VALUE to parse the JSON value. Otherwise, the parsing fails and the following error is returned:

```
2017-12-25 15:24:43,467 WARN [Topology-0 (1/1)] com.alibaba.blink.streaming.connectors.common.source.parse.DefaultSourceCollector - Field missing error, table column number: 3, data column number: 3, data field number: 1, data: [{"lg_order_code":"LP00000005","activity_code":"TEST_CODE1","occur_time":"2017-12-10 00:00:01"}]
```

- The batchGetSize value must not exceed 1,000. Otherwise, an error occurs.
- The batchGetSize parameter specifies the number of log items read at a time in a log group. If both the size of a single log item and the batchGetSize value are too large, frequent garbage collection (GC) may be triggered. To avoid this, you must set batchGetSize parameter to a smaller value.

2.15.5.2. Dimension tables

2.15.5.2.1. ApsaraDB for RDS

ApsaraDB for Relational Database Service (RDS) offers stable, reliable, and scalable cloud database services.

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
url	The URL of the RDS instance.	None
tableName	The name of the dimension table.	None
userName	The username used to access RDS.	None
password	The password used to access RDS.	None
Output Field	The fields to be returned to the descendant component.	None

Parameter	Description	Remarks
maxRetryTimes	The maximum number of retries for reading data from RDS.	None
Cache Policy	The policy for caching data.	Valid values: *None*, *LRU*, and *ALL*.
primaryKey	The primary key field of the output fields.	• You must specify the primary key when you declare the dimension table. • When you join a dimension table with another table, the ON condition must contain an equivalence condition that includes the primary key of either table. • The primary key in RDS or Distribute Relational Database Service (DRDS) is the primary key or unique index column of an RDS or DRDS dimension table.

Note

- RDS and DRDS provide the following three cache policies:
  - **None**: indicates that no data is cached.
  - **LRU**: indicates that the recently used data is cached.

    When this cache policy is used, you must set the cacheSize and cacheTTLMs parameters.
  - **ALL**: indicates that all data is cached.

  Before Realtime Compute runs a node, it loads all data in the remote table to the memory. Then Realtime Compute searches the cache for data in all subsequent dimension table query operations. In the case of a cache miss, the corresponding data does not exit. All data is cached again after the cache expires. The ALL cache policy applies to scenarios where the remote table is small but there are a large number of missing keys. When this cache policy is used, you must set the cacheTTLMs and cacheReloadTimeBlackList parameters.
- When the cache policy is set to ALL, Realtime Compute reloads data asynchronously. Therefore, you must increase the memory of the JOIN operator. The size of the increased memory is twice the data size of the remote table.
- When the cache policy is set to ALL, pay special attention to the memory of the JOIN operator to prevent out of memory (OOM) errors.

2.15.5.2.2. Table Store

Table Store is a distributed NoSQL database service built on Alibaba Cloud Apsara system. It is designed to provide high availability and data reliability.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
instanceName	The ID of the Table Store instance.
tableName	The name of the dimension table.
Output Field	The fields to be returned to the descendant component.
endPoint	The endpoint used to access Table Store. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to read data from Table Store.
accessKey	The AccessKey secret used to read data from Table Store.
Cache Policy	The policy for caching data. Valid values: **None** and **LRU**.
primaryKey	The primary key field of the output fields.

2.15.5.2.3. MaxCompute

This topic describes the parameter configuration, field type mapping, and metrics of a MaxCompute dimension table.

Parameter configuration

Parameter	Description	Remarks
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.	None
endPoint	The endpoint used to access MaxCompute. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.	None

Parameter	Description	Remarks
tunnelEndpoint	The endpoint of the Tunnel service. It corresponds to the tunnelEndPoint parameter in the WITH clause of the CREATE TABLE statement.	This parameter is required for a MaxCompute dimension table deployed in a Virtual Private Cloud (VPC).
project	The name of the MaxCompute project to which the dimension table belongs.	None
accessId	The AccessKey ID used to read data from MaxCompute.	None
accessKey	The AccessKey secret used to read data from MaxCompute.	None
Output Field	The fields to be returned to the descendant component.	None
partition	The partition name of the MaxCompute dimension table.	None
maxRowCount	The maximum number of data records that can be read from the MaxCompute dimension table	None
Cache Policy	The policy for caching data.	Default value: **ALL**.
cacheSize	The maximum number of data records that can be cached.	This parameter is required if you set the Cache Policy parameter to `LRU`. Default value: 100,000.
cacheTTLMs	The time interval at which the cache is refreshed. It corresponds to the cacheTTLMs parameter in the WITH clause of the CREATE TABLE statement. Units: milliseconds.	This parameter specifies the cache refresh interval when the Cache Policy parameter is set to `ALL`. The cache is not refreshed by default.

Parameter	Description	Remarks
cacheReloadTimeBlackList	Optional. The time period during which the cache is not refreshed. This parameter is valid when the Cache Policy parameter is set to `ALL`. During the time period specified by this parameter, for example, the Double 11 Shopping Festival, the cache is not refreshed.	This parameter is left empty by default. If you want to set this parameter, specify the time period in the format shown in the following example: `2017-10-24 14:00 -> 2017-10-24 15:00, 2017-11-10 23:30 -> 2017-11-11 08:00` Separate multiple time periods with commas (,). Separate the start and end time for a time period with the string "->".
primaryKey	The primary key field of the output fields.	None

Field type mapping

MaxCompute data type	Realtime Compute data type
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
FLOAT	FLOAT
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
DATETIME	TIMESTAMP
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
STRING	STRING
DECIMAL	DECIMAL
BINARY	VARBINARY

Metrics

When you join the dimension table to another table, you can view metrics such as the correlation degree and cache hit ratio. You can use K-Monitor to view the metrics.

Query statement	Description
fetch qps	Queries the total number of queries per second (QPS) against the dimension table, including hits and misses. The metric name is `blink.projectName.jobName.dimJoin.fetchQPS`.
fetchHitQPS	Queries the number of hits (in QPS) against the dimension table, including cache hits and hits against the physical dimension table. The metric name is `blink.projectName.jobName.dimJoin.fetchHitQPS`.
cacheHitQPS	Queries the number of cache hits (in QPS) against the dimension table. The metric name is `blink.projectName.jobName.dimJoin.cacheHitQPS`.
dimJoin.fetchHit	Queries the correlation degree of the dimension table and the table to which the dimension table is joined. The metric name is `blink.projectName.jobName.dimJoin.fetchHit`.
dimJoin.cacheHit	Queries the cache hit ratio of the dimension table. The metric name is `blink.projectName.jobName.dimJoin.cacheHit`.

Note

- We recommend that you use Realtime Compute V2.1.1 and later.
- To use a MaxCompute dimension table, you must grant the read permission to the account for accessing MaxCompute.
- When you declare a dimension table, you must specify the primary key. When you join a dimension table with another table, the ON condition must contain an equivalent condition that includes the primary key of either table.
- The primary key value for each row of a MaxCompute dimension table must be unique. Otherwise, the duplicate records are removed.
- If the dimension table is a partitioned table, Realtime Compute does not currently support writing the partition key column to the schema.
- When the cache policy is set to ALL, Realtime Compute reloads data asynchronously. Therefore, you must increase the memory of the JOIN operator. The size of the increased memory is twice the data size of the remote table.
- The following failover message may appear when you run a node:

```
RejectedExecutionException: Task
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTas,
```

Generally, this message appears because dimension table joining in Realtime Compute V1.x has certain issues. We recommend that you upgrade Realtime Compute to V2.1.1 or later. If you want to continue using the existing version, we recommend that you pause the node and resume it after troubleshooting. To troubleshoot the failover, check the specific error information that was generated for the first failover record in the failover history.

2.15.5.3. Data operators

2.15.5.3.1. Filter

The Filter component allows you to configure filter conditions. It corresponds to the WHERE clause in SQL statements.

Parameter configuration

Enter the filter expression to configure this component. The filter expression supports functions and operators (=, <>, >, >=, <, and <=), for example, `city = 'Beijing'` .

2.15.5.3.2. GroupBy

The GroupBy component corresponds to the GROUP BY clause in SQL statements.

Parameter configuration

Parameter	Description
Select grouping field	The fields based on which data is grouped. You can specify multiple fields.
Output Field	The fields to be returned, that is, the fields to be selected. You can specify the fields in the same way that you configure the Select component.

2.15.5.3.3. Join

The Join component corresponds to the JOIN clause in SQL statements.

Parameter configuration

Parameter	Description
JoinMode	The JOIN mode to be used. Valid values: INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN.
expression	The JOIN expression. An equijoin is supported, for example, leftId = rightId AND limit = 0, whereas a non-equijoin is not supported.
Select Field	The fields to be returned, that is, the fields to be selected.

2.15.5.3.4. Select

The Select component allows you to configure the fields to be returned and supports field expressions. It corresponds to SELECT statements.

Parameter configuration

Select or configure the output fields in the Select field dialog box.

You can select fields to be returned in the **Field list** section and set an alias for a field in the **Field alias** column. To set a field expression, click the Edit icon next to the target field name. In the Edit dialog box that appears, enter the required SQL statement.

2.15.5.3.5. UDTF

The UDTF component allows you to configure custom functions. It corresponds to the UDTF clause in SQL statements.

Parameter configuration

Parameter	Description
JoinMode	The JOIN mode for the custom function. Only *INNER JOIN* and *LEFT OUTER JOIN* are supported. • *INNER JOIN*: returns an empty result set when the UDTF clause returns no result. • *LEFT OUTER JOIN*: returns the NULL string when the UDTF clause returns no result.
Select function	The name of the function that the current node references. To reference a function for the current node, upload the related resources on the Resource Reference page and select the target resource.
parameter expression	The input parameters and output parameters of the referenced function.
Output Field	The fields to be returned. You can configure the name, alias, and expression of each field.

2.15.5.3.6. UnionAll

The UnionAll component corresponds to the UNION ALL clause in SQL statements.

Parameter configuration

No parameter configuration is required.

2.15.5.3.7. Dynamic column splitting

Dynamic column splitting allows you to split data records with a dynamic number of columns.

Example

Input data:

```
k1=v1,k2=v2,k3=v3,k4=v4
```

In the preceding example, the data is stored in key-value pairs in the format of key=value. Different data records may have different numbers of key-value pairs, that is, they may have different numbers of columns. In this case, you can use the first-level delimiter, which is comma (,) in the preceding example, to split the data to different key-value pairs. Then, you can use the secondary-level delimiter, which is equal sign (=) in the preceding example, to split each key-value pair to the key and value.

Parameter configuration

Parameter	Description
Select Field	The name of the field to split.
first level column delimiter	The delimiter used to split the field at the first level. Default value: \u0001.
secondary level column delimiter	The delimiter used to split the field at the secondary level. Default value: \u0002.
Add column	The fields that store the split data. Specify a key for each field. An alias is allowed.

2.15.5.3.8. Static column splitting

Static column splitting allows you to split data records with fixed columns that are separated by a fixed delimiter.

Example

You can use commas (,) as the delimiter to split the following data to four new columns, that is, 1111, 2222, 3333, and 4444.

```
1111,2222,3333,4444
```

The static column splitting method is applicable to data records with fixed columns that are separated by a fixed delimiter.

Parameter configuration

Parameter	Description
Select Field	The name of the field to split.
column separator	The delimiter used to split the field. You can use full-width characters or half-width characters as needed.

Parameter	Description
Add column	The fields that store the split data. Specify a key and a sequence number for each field. An alias is allowed.

2.15.5.3.9. Row splitting

Row splitting allows you split a row to multiple rows based on a field by using the specified delimiter.

Example

The following table lists the input data.

id	num
1	1,2

Split the row to multiple rows based on the num field by using the comma (,) as the delimiter, and place the split data in the new field new_num. The following table lists the output data.

id	num	new_num
1	1,2	1
1	1,2	2

Parameter configuration

Parameter	Description	Remarks
Select Field	The name of the field to split.	This parameter is set to num in the preceding example.
Field separator	The delimiter used to split the field. Default value: (\n).	This parameter is set to comma (,) in the preceding example.
Define new column name	The name of the new field that stores the split data.	This parameter is set to new_num in the preceding example.

2.15.5.4. Result tables

2.15.5.4.1. Datahub

Datahub is a real-time data distribution platform that is designed to process streaming data. It provides a channel for the Apsara Stack DTplus platform to process big data. Datahub works with multiple Apsara Stack services to provide an end-to-end data processing solution. Realtime Compute typically uses Datahub to store source and result tables for streaming data processing.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be returned.
endPoint	The endpoint used to access DataHub. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Datahub project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
topic	The name of the Datahub topic to which data is to be written. It corresponds to the topic parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Datahub. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access Datahub. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to DataHub. It corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.
batchSize	The number of data records that are written at a time. It corresponds to the batchSize parameter in the WITH clause of the CREATE TABLE statement.
batchWriteTimeoutMs	The interval at which the cache is cleared. It corresponds to the batchWriteTimeoutMs parameter in the WITH clause of the CREATE TABLE statement.
maxBlockMessages	The maximum number of data blocks that are written at a time. It corresponds to the maxBlockMessages parameter in the WITH clause of the CREATE TABLE statement.

Field type mapping

The following table lists the mapping between Datahub and Realtime Compute data types. We recommend that you declare the type mapping in the DDL statement.

Datahub data type	Realtime Compute data type
BIGINT	BIGINT
DOUBLE	DOUBLE
TIMESTAMP	BIGINT
BOOLEAN	BOOLEAN
DECIMAL	DECIMAL

2.15.5.4.2. Log Service

As an all-in-one real-time data logging service, Log Service allows you to quickly finish tasks such as data ingestion, consumption, delivery, query, and analysis without any extra development work. This can help you improve O&M and operational efficiency, and build up the capability to process large amounts of logs in the data technology era.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be returned.
endPoint	The endpoint used to access Log Service. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the Log Service project to which the data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field of the output fields.
source	The name of the log source. It corresponds to the source parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Log Service.
accessKey	The AccessKey secret used to access Log Service.

Parameter	Description
mode	The mode of data writing. It corresponds to the mode parameter in the WITH clause of the CREATE TABLE statement. Default value: `random`. If you set this parameter to `partition`, data is written by partition.
logStore	The name of the Logstore in the Log Service project to which the data is to be written.

2.15.5.4.3. ApsaraDB for RDS

Alibaba Cloud ApsaraDB for Relational Database Service (RDS) offers stable, reliable, and scalable cloud database services.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be written to the RDS table.
url	The URL used to access RDS. It corresponds to the url parameter in the WITH clause of the CREATE TABLE statement.
tableName	The name of the RDS table to which data is to be written. It corresponds to the tableName parameter of the WITH clause in the CREATE TABLE statement.
userName	The username used to access RDS. It corresponds to the userName parameter in the WITH clause of the CREATE TABLE statement.
password	The password used to access RDS. It corresponds to the password parameter in the WITH clause of the CREATE TABLE statement.
maxRetryTimes	The maximum number of retries for writing data to RDS. It corresponds to the maxRetryTimes parameter in the WITH clause of the CREATE TABLE statement.
batchSize	The number of data records that are written at a time. It corresponds to the batchSize parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
bufferSize	The buffer size after data deduplication. It corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement. You can only use this parameter when the primary key is defined.
flushIntervalMs	The interval at which the data cache is cleared. Unit: milliseconds. It corresponds to the flushIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
excludeUpdateColumns	The fields that will not be updated when Realtime Compute updates data records with the same primary key value. It corresponds to the excludeUpdateColumns parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip DELETE operations. It corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
partitionBy	Specifies the partitioning rule for the result table. Before writing data to the result table, Realtime Compute performs a Hash partitioning based on a partition key. The data records with the same key are then distributed to the same operator. It corresponds to the partitionBy parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field of the output fields.

Field type mapping

RDS data type	Realtime Compute data type
TEXT	VARCHAR
BYTE	VARCHAR
INTEGER	INT
LONG	BIGINT
DOUBLE	DOUBLE
DATE	VARCHAR
DATETIME	VARCHAR
TIMESTAMP	VARCHAR

RDS data type	Realtime Compute data type
TIME	VARCHAR
YEAR	VARCHAR
FLOAT	FLOAT
DECIMAL	DECIMAL
CHAR	VARCHAR

JDBC connection parameters

Parameter	Description	Default value	Since version (JDBC driver)
useUnicode	Specifies whether to use the Unicode character set. If you want to set the characterEncoding parameter to gb2312 or gbk, this parameter must be set to true.	*false*	1.1g
characterEncoding	Specifies the character encoding when the useUnicode parameter is set to true. You can set this parameter to gb2312 or gbk.	*false*	1.1g
autoReconnect	Specifies whether to automatically re-establish a connection when the connection to the database is unexpectedly interrupted.	*false*	1.1
autoReconnectForPools	Specifies whether to use the reconnection policy for a database connection pool.	*false*	3.1.3
failOverReadOnly	Specifies whether to set the connection to read-only after the database is automatically reconnected.	*true*	3.0.12

Parameter	Description	Default value	Since version (JDBC driver)
maxReconnects	Specifies the maximum number of reconnection attempts allowed if the autoReconnect parameter is set to true.	*3*	1.1
initialTimeout	Specifies the interval between two reconnection attempts if the autoReconnect parameter is set to true. Unit: seconds.	*2*	1.1
connectTimeout	Specifies the timeout period when you use a socket connection to access the database server. Unit: millisecond. The default value of 0 indicates no timeout (only works on JDK V1.4 and later).	*0*	3.0.1
socketTimeout	The timeout period for a socket operation (read or write). Unit: milliseconds. The default value of 0 indicates no timeout.	*0*	3.0.1

FAQ

- Q: When output data is written to an RDS table, is a new data record generated in the table? If not, is the result table updated based on the primary key value?

  A: If a primary key is defined in the DDL statement, the result table is updated by using the statement: `INSERT INTO tablename(field1,field2, field3, ...) VALUES(value1, value2, value3, ...) ON DUP LICATE KEY UPDATE field1=value1,field2=value2, field3=value3, ...;` . For a data record, if the value of the primary key field does not exist, the record is inserted to the table as a new row. If the value of primary key field exists, the original row in the table is updated. If no primary key is declared in the DDL statement, output data is added to the result table by using the `INSERT INTO` statement.

- Q: How can I perform GROUP BY operations based on the unique index of an RDS table?

A: An RDS table has only one auto-increment primary key, which cannot be declared as the primary key in the DDL statement of a real-time computing node. If you want to perform GROUP BY operations based on the unique index of the RDS table, declare the unique index in the Primary Key() element of the DDL statement.

2.15.5.4.4. Table Store

Table Store is a distributed NoSQL database service built on the Apsara distributed operating system of Alibaba Cloud. Based on data sharding and load balancing technologies, Table Store has high performance in scaling out and handling concurrent transactions. You can use Table Store to store and query large amounts of structured data.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
Output Field	The fields to be written to the Table Store table.
instanceName	The name of the Table Store instance. It corresponds to the instanceName parameter in the WITH clause of the CREATE TABLE statement.
tableName	The name of the Table Store table to which data is to be written. It corresponds to the tableName parameter in the WITH clause of the CREATE TABLE statement.
endPoint	The endpoint used to access Table Store. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access Table Store. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access Table Store. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
valueColumns	The names of fields to be inserted to the result table. Separate multiple names with commas (,).
bufferSize	The buffer size after data deduplication. It corresponds to the bufferSize parameter in the WITH clause of the CREATE TABLE statement.
batchWriteTimeoutMs	The timeout period for writing data to Table Store. Unit: milliseconds. It corresponds to the batchWriteTimeoutMs parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
batchSize	The maximum number of retries for writing data to Table Store. It corresponds to the batchSize parameter of the WITH clause in the CREATE TABLE statement.
retryIntervalMs	The retry interval at which data is written. It corresponds to the retryIntervalMs parameter in the WITH clause of the CREATE TABLE statement.
ignoreDelete	Specifies whether to skip DELETE operations. It corresponds to the ignoreDelete parameter in the WITH clause of the CREATE TABLE statement.
primaryKey	The primary key field of the output fields.

Field type mapping

Table Store data type	Realtime Compute data type
INTEGER	BIGINT
STRING	VARCHAR
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE

2.15.5.4.5. MaxCompute

Realtime Compute supports creating a MaxCompute table as the result table.

Parameter configuration

Parameter	Description
oriTableName	The table name used in the CREATE TABLE statement. It must be a globally unique name.
tableName	The name of the MaxCompute table to which data is to be written.
Output Field	The fields to be written to the MaxCompute table.
endPoint	The endpoint used to access MaxCompute. It corresponds to the endPoint parameter in the WITH clause of the CREATE TABLE statement.

Parameter	Description
tunnelEndPoint	The endpoint of the Tunnel service, which is required for a MaxCompute project deployed in a Virtual Private Cloud (VPC). It corresponds to the tunnelEndPoint parameter in the WITH clause of the CREATE TABLE statement.
project	The name of the MaxCompute project to which data is to be written. It corresponds to the project parameter in the WITH clause of the CREATE TABLE statement.
accessId	The AccessKey ID used to access MaxCompute. It corresponds to the accessId parameter in the WITH clause of the CREATE TABLE statement.
accessKey	The AccessKey secret used to access MaxCompute. It corresponds to the accessKey parameter in the WITH clause of the CREATE TABLE statement.
partition	The partitions to which the data is to be written. It corresponds to the partition parameter in the WITH clause of the CREATE TABLE statement. This parameter must be specified for a partitioned table. For example, if the partition name of a table is `ds=20180905`, you can specify the parameter as `` `partition` = 'ds=20180905' ``. Separate multiple levels of partitions with commas (,), for example, `` `partition` = 'ds=20180912,dt=xxxyyy' ``.

Note Realtime Compute writes cached data to a MaxCompute table every time when a checkpoint is reached.

Field type mapping

MaxCompute data type	Realtime Compute data type
TINYINT	TINYINT
SMALLINT	SMALLINT
INT	INT
BIGINT	BIGINT
FLOAT	FLOAT

MaxCompute data type	Realtime Compute data type
DOUBLE	DOUBLE
BOOLEAN	BOOLEAN
DATETIME	TIMESTAMP
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
STRING	STRING
DECIMAL	DECIMAL
BINARY	VARBINARY

FAQ

Q: Does a real-time computing node clear the result table before it writes data to the MaxCompute sink that is in Stream mode when `isOverwrite` is set to `true`?

A: The `isOverwrite` parameter is set to `true` by default. That is, a real-time computing node clears the result table and result data before it writes data to the sink. Every time a real-time computing node starts or resumes after being paused, it clears data of the existing result table or the result partition before it writes data. Certain data may be lost when data is cleared after a paused real-time computing node is resumed.

2.15.5.5. FAQ

This topic describes the frequently asked questions (FAQs) about Stream Studio.

Q: What computing engine do I need to activate before using Stream Studio?

A: You must first activate Realtime Compute because Stream Studio is a development platform based on Realtime Compute.

Q: Where can I create a Realtime Compute project? How do I bind the project to Stream Studio?

A: You can create a Realtime Compute project in the Realtime Compute console. After a project is created, you can bind it to an existing DataWorks workspace in the DataWorks console or directly create a workspace and bind the project to it. After the Realtime Compute project is bound to your workspace, you can develop real-time computing nodes in Stream Studio.

Q: What are the advantages of the directed acyclic graph (DAG) mode in Stream Studio? What are the similarities and differences between the DAG mode and SQL mode?

A: Stream Studio supports both the DAG mode and the SQL mode to develop real-time computing nodes. In DAG mode, you can perform drag-and-drop operations on components to configure real-time computing nodes without writing code. In this mode, what you see is what you get. You can also switch to the SQL mode to configure nodes by writing SQL statements.

Q: What types of SQL does Stream Studio support?

A: Realtime Compute is based on Apache Flink. Therefore, Stream Studio supports Flink SQL.

2.16. Graph Studio

2.16.1. Overview

Graph Compute is a next-generation one-stop platform for graph data management and analysis. Based on Graph Compute, Graph Studio provides an all-in-one R&D platform for Graph Compute. This topic describes how to use Graph Compute through Graph Studio.

Graph Compute allows you to model, import, and modify graph data and use the standard Gremlin language of Apache TinkerPop to query graph data. It also supports common graph analysis algorithms. Graph Compute features fast data loading, auto scaling, millisecond-level query latency, hybrid compute engines for online and offline graph computing, and shared data storage. You can use Graph Compute to easily build graph applications with large amounts of relational data.

Graph Studio provides graph application developers with all-in-one R&D services, including instance modeling, data import, data query, visualized analysis, instance management, and node O&M.

> **Note** You must bind a Graph Compute instance to your DataWorks workspace to use Graph Studio. The message shown in the following figure appears if you have not bound a Graph Compute instance to the current workspace. Bind a Graph Compute instance accordingly.

- Instance modeling: allows you to create vertices and edges to design a graph schema for your Graph Compute instance. You can configure the vertices, edges, and their attributes flexibly.
- Data import: allows you to create automatic sync nodes to import data to vertices and edges. You can schedule the sync nodes flexibly to automatically update data. Currently, you can import data stored in MaxCompute tables to Graph Compute. For more information, see Data import.
- Data query: allows you to use the standard Gremlin language of Apache TinkerPop to query graph data. For more information, see Data query.
- Node O&M: allows you to quickly filter and view the details of sync nodes. For more information, see Node O&M.

2.16.2. Instance modeling

During instance modeling, you can create vertices and edges to design a graph schema for your Graph Compute instance in visual mode or tabular mode.

Go to the Model Design page

1. Log on to the DataWorks console.
2. Click the DataWorks icon in the upper-left corner and choose **All Products > Graph Studio**.
3. On the Graph Studio page, click **Model Design**.

Visual mode

On the **Model Design** page, click the Graph Compute instance for which you want to design a schema. By default, the tab for designing the schema in **visual mode** appears.

- Create a vertex

  You can drag and drop the vertex icon to the canvas and set vertex parameters to create a vertex.

  Drag and drop the  icon to the canvas. In the right-side **Add To** dialog box that appears, set parameters as required and click **Save**.

Parameter	Description
Types	The type of the item to create. Valid values: **Point** and **Side**.
Type Name	The name of the vertex. A vertex name can contain letters, digits, and underscores (_). It can be up to 128 characters in length. **Note** The name of each vertex in the same Graph Compute instance must be unique.
Display Name	The display name of the vertex.
Remarks	The description of the vertex.
Display Color	The default color of the vertex in visual mode. Default value: *green*.
Display Size	The default size of the vertex in visual mode. Default value: *M*.
Display Content	The display content of the vertex in visual mode. Default value: the value of the **Type Name** parameter.
Attribute Configuration	You can view the name, type, description, default value, and primary key of each property of the vertex. You can click **+New Attribute** to add a property for the vertex. **Note** You can delete a property that has not been saved.

- Create an edge

  i. Move the pointer over the vertex where you want to create an edge. The  icon appears.

  ii. Drag and drop the  icon to the target vertex to connect to. In the right-side **Add To** dialog box that appears, set parameters as required.

Parameter	Description
Types	The type of the item to create. Valid values: **Point** and **Side**.

Parameter	Description
Type Name	The name of the edge. Note The name of each edge in the same Graph Compute instance must be unique.
Display Name	The display name of the edge.
Display Color	The default color of the edge in visual mode. Default value: *primary*.
Display Content	The display content of the edge in visual mode. Default value: the value of the *Type Name* parameter.
Remarks	The description of the edge.
Attribute Configuration	You can view the name, type, description, and default value of each property of the edge and perform operations on the properties. You can click **+New Attribute** to add a property for the edge.
Relationship Configuration	You must select two vertices from the created ones to set them as the source and target vertices of the edge. You can click **+Add Point-Edge Relationship** to add a vertex-edge connection for the edge as required. Note An edge can have multiple vertex-edge connections.

iii. Click **Save** to create the edge.

- Modify a vertex

Note Note the following limits when you modify a vertex:

- You cannot change the name of the vertex, but you can change the description, display color, display size, and display content of it.
- You cannot add a property as the primary key to the vertex.
- You cannot delete the primary key property from the vertex. The corresponding sync node may fail after you delete a property from the vertex. In this case, you must manually modify the node. Delete a property with caution.
- You can only change the description of existing properties for the vertex.

i. Click a vertex that you want to modify. The **Overview** dialog box appears on the right side.

ii. Click **Edit**. In the **Update** dialog box that appears, change the parameter configuration.

iii. Verify that the settings are correct and click **Save**.

- Modify an edge

> **Note** Note the following limits when you modify an edge:
>
> - You cannot change the name of the edge, but you can change the description, display color, and display content of it.
> - You can only change the description of existing properties for the edge.
> - You can add or delete properties for the edge. However, make sure that at least one vertex-edge connection exists.
> - The corresponding sync node may fail after you modify properties or vertex-edge connections of the edge. In this case, you must manually modify the node. Modify properties or vertex-edge connections with caution.

i. Click an edge that you want to modify. The **Overview** dialog box appears on the right side.

ii. Click **Edit**. In the **Update** dialog box that appears, change the parameter configuration.

iii. Verify that the settings are correct and click **Save**.

- Delete a vertex or an edge

i. Right-click a vertex or an edge that you want to delete.

ii. Select **Delete**.

iii. In the **Note** dialog box that appears, click **OK**.

Tabular mode

On the **Model Design** page, click the Graph Compute instance for which you want to design a schema. By default, the tab for designing the schema in **visual mode** appears.

Click **Tabular Mode** in the upper-right corner to switch to the tabular mode.

- Create a vertex

  In this mode, you can set relevant parameters to create a vertex.

  i. Move the pointer over the + icon and click **New Point**.

     You can also click **Add To** in the **Model Design** section to create a vertex.

  ii. In the **Add To** dialog box that appears, set parameters as required. For more information about the parameters, see the parameter description for the visual mode.

  iii. After the configuration is completed, click **Save**.

- Create an edge

  i. Move the pointer over the + icon and click **New Side**.

     You can also click **Add To** in the **Model Design** section to create an edge.

  ii. In the **Add To** dialog box that appears, set parameters as required. For more information about the parameters, see the parameter description for the visual mode.

  iii. After the configuration is completed, click **Save**.

- Modify a vertex

  Click **Change Settings** in the Actions column of a vertex that you want to modify. In the **Update** dialog box that appears, change the parameter configuration as required and click **Save**.

> **Note** Note the following limits when you modify a vertex:
>
> - You cannot change the name of the vertex, but you can change the description, display color, display size, and display content of it.
> - You cannot add a property as the primary key to the vertex.
> - You cannot delete the primary key property from the vertex. The corresponding sync node may fail after you delete a property from the vertex. In this case, you must manually modify the node. Delete a property with caution.
> - You can only change the description of existing properties for the vertex.

- Modify an edge

  Click **Change Settings** in the Actions column of an edge that you want to modify. In the **Update** dialog box that appears, change the parameter configuration as required and click **Save**.

> **Note** Note the following limits when you modify an edge:
>
> - You cannot change the name of the edge, but you can change the description, display color, and display content of it.
> - You can only change the description of existing properties for the edge.
> - You can add or delete properties for the edge. However, make sure that at least one vertex-edge connection exists.
> - The corresponding sync node may fail after you modify properties or vertex-edge connections of the edge. In this case, you must manually modify the node. Modify properties or vertex-edge connections with caution.

2.16.3. Data import

Currently, Graph Studio only allows you to import data stored in MaxCompute tables to Graph Compute.

Go to the Data Import page

After you create a vertex or an edge, a corresponding sync node is automatically generated for the vertex or edge. You can import data stored in MaxCompute tables to the vertex or edge in **visual mode** or **tabular mode**. Follow these steps to open the data import page in the visual mode and tabular mode respectively:

- **Visual mode**

  Open the Graph Compute instance. The tab for importing data in **visual mode** appears by default, where you can import data to vertices and edges.

  - Import data to a vertex: Right-click a vertex and select **Data Import**.
  - Import data to an edge: Right-click an edge, select **Data Import**, and then click a vertex-edge connection.

- **Tabular mode**

  Open the Graph Compute instance. The tab for importing data in visual mode appears by default. Click **Tabular Mode** in the upper-right corner switch to the tabular mode. Click **Data Import** in the Actions column of a vertex or an edge to import data.

Configure the sync node

After you select or click **Data Import** for a vertex or an edge, the **DataStudio** page appears.

- Import data to a vertex
  i. Configure the source and destination connections for the sync node.

     Set Connection in the **Source** section to ODPS and specify the MaxCompute table and partitions from which data is imported. The Connection parameter in the **Target** section is set to the target vertex of the Graph Compute instance by default.

  ii. Fields in the source table on the left have a one-to-one mapping with fields in the destination table on the right. You can click **Add** to add a field or move the pointer over a field and click the **Delete** icon to delete the field.

  iii. Configure the channel.

Parameter	Description
Expected Maximum Concurrency	The maximum number of concurrent threads to read and write data to data storage within a single sync node. You can configure the concurrency for a node on the codeless UI.
Bandwidth Throttling	Specifies whether to enable bandwidth throttling. You can enable bandwidth throttling and set a maximum transmission rate to avoid heavy read workload of the source. We recommend that you enable bandwidth throttling and set the maximum transmission rate to a proper value.
Dirty Data Records Allowed	The maximum number of dirty data records allowed.
Resource Group	The servers on which nodes are run. If an excessively large number of nodes are run on the default resource group, some nodes may be delayed due to insufficient resources. In this case, you can add a custom resource group.

  iv. Configure the node properties.

     Click the **Properties** tab in the right-side navigation pane. On the **Properties** tab that appears, set the parameters. For more information, see Schedule.

  v. Commit the node.

     After you set the properties, click **Save** in the tool bar to commit the node to the development environment. After you commit the node to the development environment, the node is unlocked.

  vi. Deploy the node.

     For more information, see Deploy.

  vii. Test the node in the production environment.

     For more information, see Recurring tasks.

- Import data to an edge

The procedure of importing data to an edge is similar to that of importing data to a vertex. The difference is that you must specify a target vertex-edge connection to which data is imported. In addition, you must specify the primary keys of the source and destination vertices connected by the edge For more information, see Import data to a vertex.

> **Note** If you want to delete a vertex or an edge from a schema, you must also manually delete it from the sync node.

2.16.4. Data query

Graph Studio allows you to use the graph traversal language, Gremlin, of Apache TinkerPop to query graph data.

Create a graph query node

1. Go to the Graph Studio page. In the left-side navigation pane, click **Data Query**.
2. On the Data Query page, move the pointer over the **Create** icon and choose **Create > Graph Query**.
3. In the **Create Node** dialog box that appears, set the **Diagram Example**, **Location**, **Node Name** parameters.
4. Click **Submit** to create the graph query node.
5. Enter and run the Gremlin query statement in the editor. The query result is returned.

View the query results

The following figure shows the query results of running the `g.V().limit(10)` statement.

As shown in the preceding figure, the results section provides three tabs to display the query results in graph mode, display the query results in tabular mode (in the form of an Excel file), and display operational logs, respectively. By default, operational logs appear when you run the graph query node. If the query results contain vertices or edges, they are provided in graph mode. Otherwise, the query results are provided in tabular mode.

- Graph mode

  In graph mode, vertices or edges are provided in a graph so that you can clearly view them. You can click a vertex or an edge to view its detailed information and properties.

- Tabular mode

  In tabular mode, the detailed information and properties of each vertex or edge are provided in a table.

- Operational logs

  The log tab displays the operational logs generated when you run the Gremlin statement in the graph query node.

2.16.5. Node O&M

The Operation and Maintenance page displays all sync nodes of a Graph Compute instance, so that you can view the data update status.

Click **Operation and Maintenance** in the upper-right corner to go to the corresponding page. You can also filter sync nodes by instance name, vertex or edge name, running status, node type, and data timestamp to find the required sync node and view its details.

2.17. Data Protection

2.17.1. Overview

Data Protection is a data security management platform. It can be used to detect data assets, detect sensitive data, classify data, de-identify data, monitor data access behavior, report alerts, and audit risks.

Data Protection provides security management services for MaxCompute.

Access Data Protection

1. Log on to the DataWorks console.
2. On the DataWorks page that appears, click the icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the Data Protection page.

Features

Data Protection provides the following features:

- Intelligent sensitive data detection

  Data Protection automatically detects an enterprise's sensitive data based on self-training models and algorithms, and clearly displays statistics on data types, volume, and visitors. It also recognizes custom data types.

- Accurate data classification: Data Protection allows you to classify data and create custom levels for better data management.
- Flexible data de-identification

  Data Protection provides diverse and configurable methods for dynamic data de-identification.

- Risky behavior monitoring and auditing

  Data Protection uses various correlation analysis algorithms to detect risky behavior. It also provides alerts and supports visualized auditing for detected risks.

2.17.2. Configure rules for defining sensitive data

This topic describes how to configure rules for defining sensitive data.

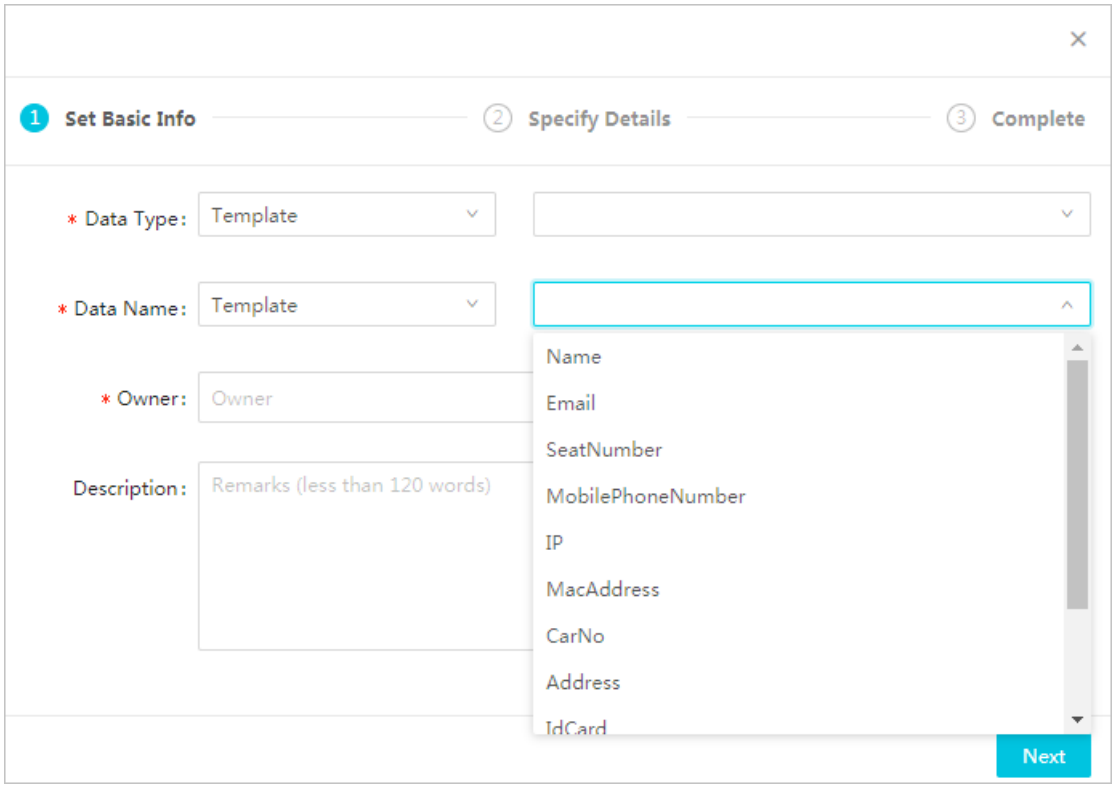
Procedure

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Data Definition**. On the Data

Recognition Rules page that appears, click **Create Rule**.

5. In the dialog box that appears, set parameters in the **Set Basic Info** step.

   You can create a template-based identification rule or a custom identification rule.



Parameter	Description
Data Type	The category of the rule. You can select Template or Custom from the Data Type drop-down list. ○ If you select **Template**, you can select **Personal Information**, **Merchant Information**, or **Company Information** from the right drop-down list. ○ If you select **Custom**, you can enter a data type.
Data Name	○ If you select **Template** from the drop-down list, you can select a built-in identification rule template from the right drop-down list. You can select **Email**, **SeatNumber**, **MobilePhoneNumber**, **IP**, **MacAddress**, **CarNo**, **PostCode**, **IdCard**, or **BankCard**. ○ If you select **Custom**, you can enter a data name.
Owner	The owner of the rule.
Description	The description of the rule.

6. Click **Next**. Set the **Level** and **Data Definition** parameters.

Parameter	Description
Level	The security level of the sensitive data to which the rule is applied. If the existing security levels cannot meet your needs, click **Levels** in the left-side navigation pane to change the level settings.
Content Scanning	Specifies whether to enable content scanning. This option is selected by default for all the built-in data identification templates. ○ If you select a template, you cannot modify the identification rule, but you can verify the accuracy of the identification rule. ○ If you select regular expression matching, you can customize the identification rule.
Field Scanning	Specifies whether to enable field scanning. This approach provides two matching methods: exact matching and fuzzy matching of field names. Multiple-field matching is supported, and the relationship between the fields is OR.

7. Click **Next**. After you confirm the configuration, click **Save and Apply**.

> **Note** When you create a rule to define sensitive data, note the following:
> - The rule name must be unique.
> - The content scanning and field scanning configuration must be unique.
> - You can only view the sensitive data that is detected based on the data identification rule one day after the rule takes effect.

2.17.3. View the distribution of sensitive data

On the next day after you configure and activate sensitive data identification rules as a data security administrator, you can access Data Recognition to view the distribution of sensitive data.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. In the left-side navigation pane, click **Data Recognition**. On the Data Recognition page that appears, you can view the overall data distribution and field details.

2.17.4. View the information about data activities

On the next day after you configure and activate sensitive data identification rules as a data security administrator, you can access Data Activities to view related activity statistics, trend, and details.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. In the left-side navigation pane, click **Data Activities** to go to the **Data Activities** page.

The Data Activities page allows you to view the information of each activity that involves sensitive data. On the Manipulations and Queries tab, you can view the statistics, trend, user, and details of data access activities. On the Export tab, you can view the statistics and details of data export.

2.17.5. View the data audited as risky

Data activities are audited manually or based on the risk identification rules and AI-based identification rules. The Data Risks page displays data activities that are audited as risky. You can comment audit results as required.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, click **Data Risks** to filter and view the data audited as risky as needed.

2.17.6. Manage the data security levels

When creating a rule, you can specify a security level for the data to which the rule applies. On the Levels page, you can create and delete security levels. You can also modify the priority of each security level and manage rules by security level.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Levels**.

   On the **Levels** page, you can create and delete security levels. You can also modify the priority of each security level and manage rules by security level.

Operation	Description
Create a security level	Click **Create Level**. Specify the security level name and operator.
Manage rules by security level	Find the target security level and click the  icon in the Actions column. In the Manage Rules by Level dialog box that appears, you can select a rule and adjust its security level.
Delete a security level	Find the target security level and click the  icon in the Actions column. In the dialog box that appears, click **Delete**.
Modify the priority of a security level	Find the target security level. Drag and drop the  icon in the Actions column.

2.17.7. Manage data that is incorrectly detected

On the Manual Check page, you can manually correct the sensitive data that is incorrectly detected by rules. For example, you can delete incorrectly detected data, change the type of the detected data, and delete or recover data in batches.

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now** to go to the **Data Security Guard** page.
4. In the left-side navigation pane, choose **Management > Manual Check**.

   On the Manual Check page, you can delete incorrectly detected data, change the type of the detected data, and delete or recover data in batches.

   - To delete a data record that is incorrectly detected, turn off the switch in the Status column of the data record.

     > **Note** You can recover data records that you have deleted.

   - To change the type of a data record, click the edit icon next to the name of the target rule and select a rule.

     > **Note** You can only select a rule that has been configured in DataWorks.

   - To delete or recover multiple data records at the same time, you can select the data records and click **Remove** or **Recover**.

2.17.8. Customize de-identification rules

This topic describes how to customize de-identification rules in Data Security Guard so that DataWorks can dynamically de-identify the results of ad hoc queries.

Prerequisites

Sensitive data detection rules are created and data security levels are specified. For more information, see Configure rules for defining sensitive data and Manage the data security levels.

Go to the Data Masking page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now**.
4. In the left-side navigation pane, choose **Management > Data Masking**.

   The **Data Masking** page has two tabs: **Data Masking** and **Whitelist**.

Customize de-identification rules in Data Security Guard

1. Set the **Masking Scene** parameter to **Global Config (_default_scene_code)** and click **Create**

Rule in the upper-right corner.

2. In the **Create Rule** dialog box, set the **Rule**, **Owner**, and **Method** parameters.

   > **Note** Data Security Guard provides three methods for de-identifying ID card numbers and email addresses, including **Pseudonymisation**, **Hashing**, and **Masking Out**. For other types of data, Data Security Guard only provides the **Hashing** and **Masking Out** methods.

   - **Pseudonymisation**

     This method replaces the text of a data record with an artificial pseudonym of the same data type. If you select this method, specify a security domain. Rules with different security domains generate different pseudonyms for the same data record.

   - **Hashing**

     If you select this method, specify a security domain. Rules with different security domains generate different hash values for the same data record.

   - **Masking Out**

     This method uses asterisks (*) to mask specified parts of a data record. It is commonly used.

Parameter	Description
Recommended	You can select recommended policies to mask data of common types such as ID card numbers and bank card numbers.
Custom	You can flexibly specify whether to mask the specified number of characters at the first, middle, or last part of a data record.

3. Click **OK**.
4. On the **Data Masking** tab of the **Data Masking** page, set the status of the de-identification rule to **Active** or **Inactive**.

   You can click the **Test** icon in the Actions column of the rule to test whether it works.

5. Click the **Whitelist** tab. On the Whitelist tab, click **Add Account**.
6. In the **Add Account** dialog box, set the **Rule**, **Account**, **Effective From**, and To parameters. For more information about user groups, see Manage user groups.

   > **Note** If you query data beyond the time range specified for the whitelist, the query results will be de-identified.

7. Click **Save**.

Verify the de-identification effect in DataWorks

After you create and configure de-identification rules, DataWorks dynamically de-identifies the results of queries in your workspace based on the rules.

> Note You must first turn on Mask Data in Page Query Results for your workspace in the DataWorks console.

2.17.9. Manage user groups

You can create a user group on the GroupManagement page and reference it in a de-identification whitelist. You can also copy, edit, and delete user groups on the GroupManagement page.

Go to the GroupManagement page

1. Log on to the DataWorks console.
2. On the DataStudio page, click the DataWorks icon in the upper-left corner and choose **All Products > Data Protection**.
3. Click **Try now**.
4. In the left-side navigation pane, choose **Management > GroupManagement**. The GroupManagement page appears.

Create a user group

1. On the **GroupManagement** page, click **Create Group** in the upper-right corner.
2. In the **Create Group** dialog box that appears, set the parameters described in the following table.

Parameter	Description
Name	Enter the name of the user group. > Note The name of the user group must be unique.
Owner	Enter the owner of the user group.
Source Type	Specify the source of accounts in the user group. Valid values: ○ **Text**: If you select this option, click **Upload File** next to **Source File**, select a local file to upload, and then click **Open**. ○ **Select Existing Accounts**: If you select this option, select the accounts to add next to **Add Members** and click >.

3. Click **Save**.

Copy a user group

On the **GroupManagement** page, find the target user group and click  in the Actions column.

An identical user group is generated.

> Note
>
> - The name of the generated user group contains the -copy suffix. You can click to change the name.
> - You can only copy the content but not the dependencies of a user group.

Edit a user group

To edit an existing user group, follow these steps:

1. On the **GroupManagement** page, find the target user group and click in the Actions column.
2. In the **Edit Group** dialog box that appears, modify parameters such as **Name**, **Owner**, and **Source Type**.
3. Verify the settings and click **Save**.

Delete a user group

To delete a user group, find the user group on the GroupManagement page and click **Delete** in the Actions column. In the dialog box that appears, click **Delete**.

> Note You cannot delete a user group that is referenced in a de-identification whitelist.
>
> If you still want to delete the user group, delete the user group from the corresponding de-identification whitelist first.

2.18. App Studio

2.18.1. Overview

App Studio is a tool designed to help you develop data products. It comes with a rich set of front-end components that you can drag and drop to simply and quickly build front-end apps.

With App Studio, you do not need to download and install a local integrated development environment (IDE) or configure and maintain environment variables. Instead, you can use a browser to write, run, and debug apps and enjoy the same programming experience as that in a local IDE. App Studio also allows you to publish apps online.

Advantages

App Studio has the following core advantages:

- Data development anytime, anywhere

  You do not need to download and install a local IDE or configure and maintain environment variables. Instead, you can use a browser to develop data in your office, at home, or anywhere that you can connect to the network.
- Editor with complete features

App Studio provides a browser-based editor that allows you to easily write, run, and debug projects. When you enter the code, App Studio provides code hinting, code completion, and repair suggestions. You can also find all references and the definition of a method to automatically generate code.

- Online debugging

  App Studio comes with all breakpoint types and operations of a local IDE. It supports thread switching and filtering, variable checking and watching, remote debugging, and hot code replacement.

- Multi-feature terminal

  You can directly access the runtime environment, which is currently built based on CentOS as the base image. The multi-feature terminal supports all bash commands, including vim and other interactive commands.

- Collaborative coding

  You and your team members can use App Studio to share the development environment for collaborative coding. Currently, App Studio allows a maximum of eight users to edit the same file of a project online concurrently, improving work efficiency. In the future, the collaborative coding component will support chatting, bullet screen messages, code annotations, videos, and other features to make teamwork efficient and pleasant.

- Plug-in system

  App Studio supports business plug-ins, tool plug-ins, and language plug-ins.

  - App Studio allows you to customize any required menu or add any service portal based on your business needs.
  - You can customize project management processes, project types, and templates dedicated to your business.
  - You can develop common tools, such as enhanced Git features, code rule scanning, keyboard shortcuts, enhanced editing features, and code snippets, and integrate them into App Studio.
  - You can use language plug-ins to enrich the languages supported by App Studio, enabling App Studio to serve users with more languages while addressing your own business needs.

- Visual building

  App Studio provides a WYSIWYG designer that has rich components and deeply integrates DataService Studio and DataStudio. Among all components of DataWorks, you can call DataWorks API operations only in App Studio. In addition to calling the API operations, you can quickly build front-end apps by dragging and dropping components and configuring them in the WYSIWYG designer based on the santa file system, developing web apps without code.

- Rich templates and flexible project management

  App Studio provides rich project templates, allowing you to develop your project accordingly with fewer steps and higher efficiency. You can also save your project as a template for future development and use, or share it with other users.

2.18.2. Get started with App Studio

To build a data portal, engineers need to develop data, build backend services, and develop front-end pages. This topic describes the basic features of App Studio and how to use App Studio.

Originally, DataWorks is mainly used by data engineers to implement offline or streaming data development. As DataWorks becomes increasingly easy to use, many roles such as algorithm engineers, BI analysts, operators, and product managers who are familiar with SQL can use DataWorks to develop data.

App Studio helps different types of users quickly build webpages for data viewing and apps for data query.

Go to the App Studio page

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products** > **App Studio**. The **Projects** page appears.

Create a front-end project

App Studio provides complete front-end development capabilities that allow you to develop front-end projects in the same way as in a local integrated development environment (IDE). Without the need to master or understand any new concepts, you can create front-end projects in App Studio and develop HTML, CSS, JavaScript, and React files in a way that you are familiar with.

1. Create a project based on the sample project.
   i. Go to the **App Studio** page and click **Projects** in the left-side navigation pane. On the Projects page, click **Create Project from Code**.
   ii. On the **Create Project** page, set the **Name** and **Description** parameters, and set the runtime environment to **react-demo**.

   > Note
   > - The name of the project must start with a letter and can contain digits, letters, underscores (_), and hyphens (-).
   > - The description of the project can be 2 to 500 characters in length.

   iii. After the configuration is completed, click **Submit**.
2. Set running parameters.

   In the upper-right corner, choose **Edit Config** > Edit Configurations. In the **Run/Debug Configurations** dialog box that appears, set the required parameter. Select the instance type and specify the port number as required. You can use the default configuration unless otherwise required. Then, click **OK**.

Parameter	Description
Install Cmd	The command used to install the dependency, for example, npm install.
Start Cmd	The command used to start the app, for example, npm start.

Parameter	Description
Environment Variables	The environment variables.
Initialize Script	The path of the script used to initialize a container in the code library.
PORT	The port of the Elastic Compute Service (ECS) instance. Default value: 3000.
ECS Instance	The instance type. Valid values: 1vCPU 2GMemory, 2vCPU 3GMemory, 4vCPU 8GMemory, and 8vCPU 16GMemory.

3. Run the project.

   Click the Run icon in the upper-right corner to run the project. Currently, you can run the `tnpm start` command to start front-end projects. You can seamlessly run projects with *webpack-dev-server* configured.

   During project running, you can view the dependency installation and app startup logs. After the project running is completed, the Preview tab appears in the right-side navigation pane. You can edit and save the code in real time. The edited code takes effect immediately.

4. Access the project.

   Click the **Preview** tab in the right-side navigation pane, and click the arrow next to the access link to open the project.

   In App Studio, you can edit and develop front-end projects in the same way as in a local IDE. App Studio supports code completion, method signature, refactoring, and redirection for HTML, CSS, LESS, SCSS, JavaScript, TypeScript, JSX, and TSX files. In addition, you can develop front-end projects based on templates without the need to build any environment or download any dependency.

Create a backend project

1. Create a project based on the sample project.
   i. Go to the **App Studio** page and click **Projects** in the left-side navigation pane. On the Projects page, click **Create Project from Code**.
   ii. On the **Create Project** page, set the **Name** and **Description** parameters, and set the runtime environment to **springboot**.
   - The name of the project must start with a letter and can contain digits, letters, underscores (_), and hyphens (-).
   - The description of the project can be 2 to 500 characters in length.

   iii. After the configuration is completed, click **Submit**.
2. Set running parameters.

   In the upper-right corner, choose **Edit Config** > Edit Configurations. In the **Run/Debug Configurations** dialog box that appears, set the required parameter and then click **OK**.

Parameter	Description
Main class	Select the main method. If no main method is available, check whether your project has a main method.
VM options	The virtual machine (VM) options.
Program arguments	The app parameters.
Environment Variables	The environment variables.
JRE	The Java runtime environment (JRE). By default, this parameter cannot be modified.
PORT	The port of the ECS instance. Default value: *7001*.
ECS Instance	The instance type. Valid values: **1vCPU 2GMemory**, **2vCPU 3GMemory**, **4vCPU 8GMemory**, and **8vCPU 16GMemory**.
Pre-Launch Option	The commands to be run before the project is run. You can specify up to three commands.
Enable Hot Code	Specifies whether to enable hot code replacement.

You can click **Add** on the left of the Run/Debug Configurations dialog box to add multiple configurations for running.

3. Run the project.

   Click the Run icon in the upper-right corner to run the project.

   The first time that the project is run takes a longer time because App Studio needs to allocate the ECS instance and initialize the language service. After the running is completed, the Runtime tab appears, showing the access link.

4. Access the project.

   Click **Open Link** to access the project.

Append /testapi to the link and refresh the page.

Understand App Studio

The following operations are supported for created projects:

- Top navigation bar
  - **Project**

    From the Project menu, you can configure the project or view detailed information by selecting **Character Set** or **Project Information**. Provided information about the current project includes the ID specified by **Project ID**, name specified by **Project Name**, type specified by **Project Type**, creation time specified by **Created At**, and **UUID**.

  - **File**

    From the File menu, you can create a file or open a recently created file by selecting **Create File** or **Re-Open Most Recent Files**.

  - **Edit**

    From the Edit menu, you can perform common editing operations. To search all the code in the project and open the related file, select Find in Path.

- **Version**

  From the Version menu, you can select **Switch Branch**, **View Changes**, **Submit**, **View Log**, **Connect to Remote Repo**, and **Merge Abort**.

  - **Switch Branch**

    In the Check Out Branch dialog box, you can click **+Create Branch** to create a local branch and push it to the remote repo. You can click a local branch and select **checkout** from the shortcut menu on the right to switch to the branch. You can also select **merge** to merge the selected branch to the current branch.

    You can click a remote branch and select **check out as a new local branch** from the shortcut menu on the right to check out the remote branch locally. Then rename the branch. You can also select **merge** to merge the selected branch to the current branch.

  - **View Changes**

    Click **View Changes** to view the list of edited files on a local branch in the right-side navigation pane.

  - **Submit**

    Click Submit to commit edits on a local branch for staging. You must enter the commit information.

  - **View Log**

    On the Log page, you can view all commit records of branches and filter them.

  - **Connect to Remote Repo**

    You can associate a new project with a remote repo for version control.

- **View**

  You can click **Toggle Full Screen** or press **Esc** on the keyboard to enter or exit the full screen mode of the page. You can also click **Hide Sidebar** or **Hide Status Bar** to hide the right-side navigation pane or the status bar. If they are hidden, you can click Show Sidebar or Show Status Bar to show them respectively.

- **Debug**

  - If you create a front-end project, you can set running parameters and add custom images.
  - App Studio supports Java-based debugging. In addition to setting running parameters and adding custom images, you can perform many other operations for debugging backend projects. You can also perform full or incremental builds and compile the Main.java file.

- **Settings**

  From the Settings menu, you can set the Git configuration to import the Git code to create a project. You can also configure your preference and shortcut keys.

- **Deploy**

  You can choose **Deploy** > **Download Source Code** to download the source code.

- **Template**

  You can choose **Template** > **Manage Templates** to go to the **My Templates** page to manage templates.

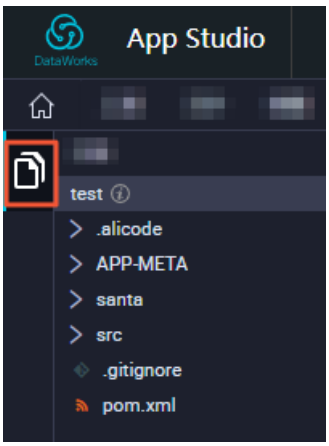
- Left-side navigation pane

- Entry

  Click the icon framed in red. The project section appears.

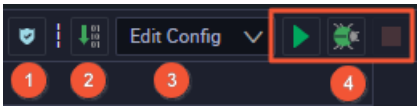
- Edit section

  Double-click a file that you want to edit. In the Edit section that appears, right-click the code section to perform the following operations.



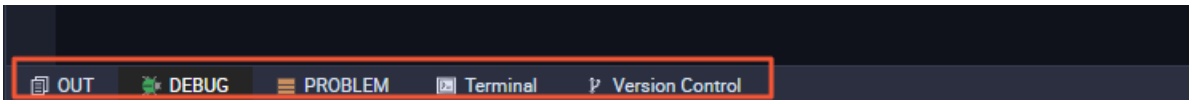
Action	Description
Go to Definition	Navigates to the definition page.
Peek Definition	Previews the definition.
Find All References	Searches for all references.
Workspace Symbol	Searches for a symbol in the project.
Go to Symbol...	Navigates to the symbol in the project.
Generate...	Generates the code.
Rename Symbol	Renames the symbol.
Change All Occurrences	Changes the name of all occurrences of a symbol throughout the file.
Format Document	Formats the file.
Cut	Cuts the file.
Copy	Copies the file.
Command Palette	Goes to the command palette.

- Icons in the upper-right corner



No.	Feature
1	Alibaba Coding Guidelines
2	Build Program. You can perform this operation only when the project is running or being debugged.
3	Run/Debug Configurations. You can set parameters for running or debugging the project.
4	Operations on the project, including running, debugging, or stopping the project.

- Bottom bar



  - OUT tab

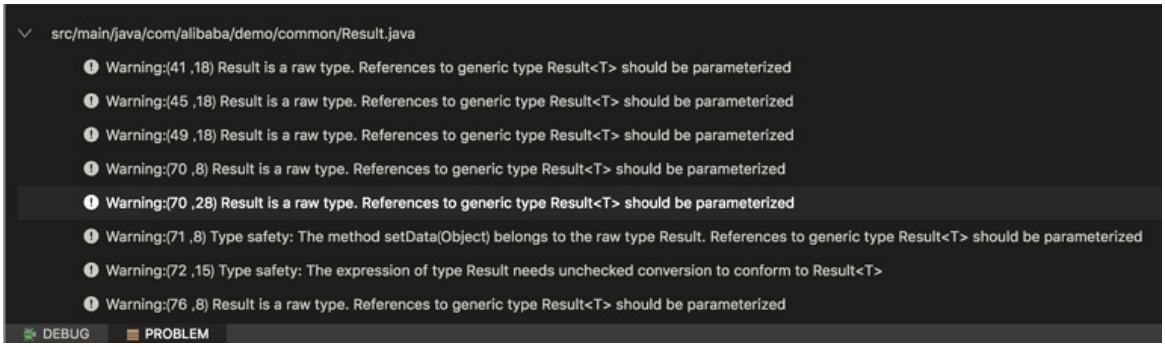
    You can click the OUT tab to view the output.

  - RUN or DEBUG tab

    If you click the Run or Debug icon for a project, this tab appears, showing the progress and information of the project.

  - PROBLEM tab

    If you click the Run or Debug icon for a project that has a problem, this tab appears.



- Terminal tab

  When running or debugging a project, you can click the Terminal tab and run bash or vim commands on the ECS instance.

- Version Control tab

  You can click the Version Control tab to view the logs and history of the project.

2.18.3. Navigation pane

2.18.3.1. View and manage projects

You can create and manage projects on the Projects page.

Go to the App Studio page and click Projects in the left-side navigation pane. On the page that appears, you can view projects that you have created. For more information about how to create template-based and code-based projects, see Project management.

Click a project to go to the project editing page. You can also click **Create Template** of a project to create a template based on the project.

Create a template

1. Click **Create Template** of a project.
2. In the **Create Template** dialog box that appears, set each parameter.

Parameter	Description
Name	The name of the template.
Description	The description of the template.
Class	The class of the template.

3. After the configuration is completed, click **OK**.

2.18.3.2. View and manage templates

You can view all templates created based on projects on the Templates page.

Click a template to go to the template details page. Then, click **Code Editor** to view the project code that this template is based on.

You can also click **Create Project** of a template to create a project based on this template.

2.18.4. Project management

This topic describes how to create and manage projects.

You can create a template-based or code-based project.

Create a template-based project

1. Go to the App Studio page and click Projects in the left-side navigation pane. On the **Projects** page, click **Create Project from Template**.
2. On the **Create Project** page, specify **Name** and **Description**, and select a template.

   > **Note**
   > - You can select a custom template or a template provided by the system.
   > - All projects created by using templates support WYSIWYG development.

3. After the configuration is completed, click **Submit**.

Create a code-based project

You can create a project by running code. App Studio provides code templates for three types of runtime environments. Select a code template as required.

1. Go to the App Studio page and click Projects in the left-side navigation pane. On the **Projects** page, click **Create Project from Code**.
2. On the **Create Project** page, specify **Name** and **Description**, and select a template.
3. After the configuration is completed, click **Submit**.

View and manage projects

You can view the created projects on the **Projects** page.

You can click a project name to go to the project editing page. You can also click **Create Template** of a project to create a template based on the project.

> **Note** You can view projects shared by others but cannot create templates based on those projects.

2.18.5. Code editing

2.18.5.1. Overview

Code editing supports common IDE features, such as automatic completion, code hinting, syntax diagnosis, and global content search.

The following tables list the basic and advanced features that App Studio supports in different languages.

Basic feature	Java	Python	JavaScript and TypeScript
Completion	Supported	Supported	Supported
Hover	Supported	Supported	Supported
Diagnostics	Supported	Supported	Supported
SignatureHelp	Supported	Supported	Supported
Definition	Supported	Supported	Supported
References	Supported	Supported	Supported
Implementation	Supported (coming soon)	Not supported	Not supported
DocumentHighlight	Supported	Supported	Supported
DocumentSymbol	Supported	Supported	Supported
WorkspaceSymbol	Supported	Supported	Supported
CodeAction	Supported (Alibaba Java Guidelines coming soon)	Supported	Supported
CodeLens	References implementation	Not supported	Not supported
Formatting	Supported	Supported	Not supported
RangeFormatting	Supported	Not supported	Not supported
FindInPath	Supported	Supported	Supported

Advanced feature	Java	Python	JavaScript and TypeScript
Rename	Supported	Supported	Supported
WorkspaceEdit	Supported	Not supported	Not supported
UnitTest (quick start)	Supported	Not supported	Not supported
MainClass	Supported	Not supported	Not supported
MainClassQuickStart	Not supported	Not supported	Not supported
ListModules	Supported	Not supported	Not supported

Advanced feature	Java	Python	JavaScript and TypeScript
Generate	Constructor Override Getter and Setter Implement	Not supported	Not supported

2.18.5.2. Generate code snippets

Currently, App Studio supports the Java class constructor, getter and setter methods, override methods of the parent class that a child class inherits, and API methods to be implemented.

Entry

Perform either of the following operations to generate the Java code:

- Right-click the code section and select **Generate**.
- Press Command+M on the keyboard. The Java code is automatically generated.

Constructor

On the Generate menu, click **Constructor**.

Select the fields to be included in the constructor and click **OK**.

The constructor that contains the initialization statement of the fields is generated.

Getter and setter methods

Generate the getter and setter methods in a way similar to the constructor.

> **Note** If a Java class does not have any field or the Java class is overwritten by the @data annotation of lombok, the getter or setter method is not required for the Java class. In this case, the Getter, Setter, and Getter And Setter options do not appear on the Generate menu.

Override methods

Click Override Methods on the Generate menu. All methods that can be overridden are listed in the Generate Code dialog box.

Select a method. The corresponding method is generated.

2.18.5.3. Run UT

App Studio currently supports unit testing (UT), including automatically generating UT code, detecting the entry for UT, running UT code, and displaying the UT result.

Automatically generate UT code

Open the target file, right-click the code editing section, select **Generate** and then click **Create Test**. The UT class file and UT code are automatically generated in the test directory.

Detect the entry for UT

> Note
>
> - UT class files must be stored in the *src/test/java* directory. A Java UT class file that is not stored in this directory cannot be identified as the Java UT class.
> - For a method annotated with @Test annotation, Run Test appears, indicating the entry for UT.

After the Java UT class file is created, add the **@Test** annotation of `org.junit.Test` to the corresponding sample UT method.

Run UT code

Click the Run icon in the upper-right corner. The sample UT starts.

2.18.5.4. Find in Path

App Studio provides the Find in Path feature to support global content search.

Move the pointer over **Edit** in the top navigation bar and select **Find in Path**.

You can select **Match Case**, **Words**, **Regex**, and **File Mask** as required. If you select File Mask, you must also select a file name extension from the right drop-down list to search in files of the specified type.

You can also search for content in the specified project, module, or directory.

After selecting a file, you can locate the searched content in the file and open the file in the editor.

2.18.6. Debugging

2.18.6.1. Configuration and startup

You can configure the entry method, start debugging, and set breakpoints to debug an app.

Configure the entry method

Parameter	Description
Main class	The entry method (which is the main method) you want to start. You can select a value from the drop-down list.
VM options	The parameters for starting a Java Virtual Machine (JVM), for example, -D, -Xms, and -Xmx.
Program arguments	The startup parameter, which is obtained by the args parameter in the main method.

Parameter	Description
Environment Variables	The environment variables.
JRE	The Java runtime environment. Default value: *1.8 - SDK*.
PORT	The port you want to expose in the app, for example, classic port 7001 or port 8080 for Spring Boot-based projects.
ECS Instance	The type of the ECS instance used for debugging.
Enable Hot Code	This configuration takes effect only in Run mode. By default, the HotCode2 plug-in that Alibaba Cloud provides is used.

Start debugging

Move the pointer over **Debug** in the top navigation bar and click **Start Debugging**.

The first startup is slower, because the system needs to prepare the runtime environment and download Maven dependencies for you. When you restart debugging, App Studio skips this process and provides user experience similar to that in a local IDE.

2.18.6.2. Online debugging

App Studio supports the online debugging of Java apps and Spring Boot-based web projects.

Before online debugging, you must configure the entry method and start debugging. For more information, see Configuration and startup.

Exposed services

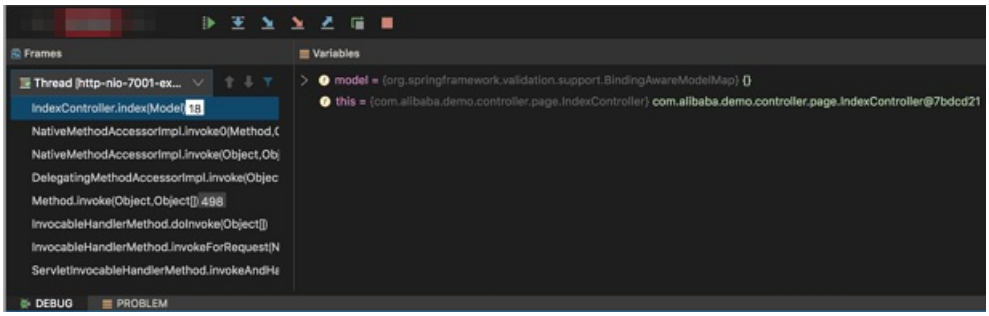
After your app is started, two basic services are provided. You can click the link next to Backend to debug the back-end Java code.

Panel introduction

- Output

  The Output panel displays the standard output, excluding System.in, of all apps. It supports the ANSI color and guarantees consistent experience as a local terminal.

- Call Stack

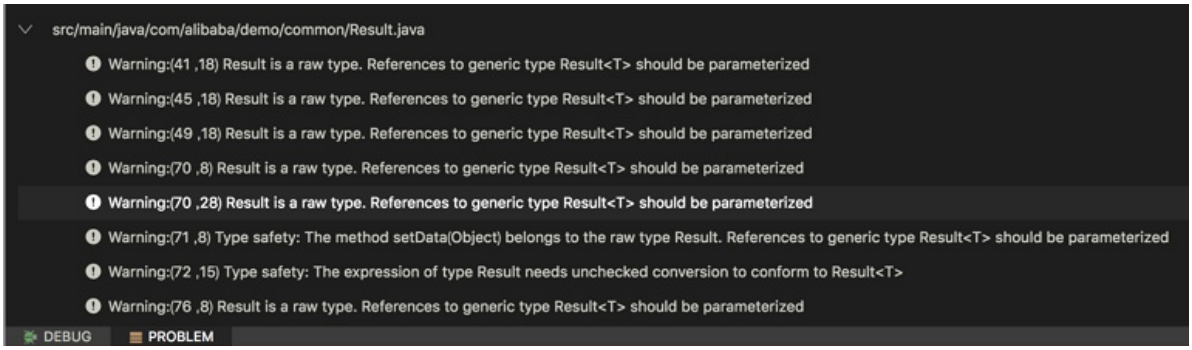


- Breakpoint

  The Breakpoint panel displays the breakpoints that are currently set. For more information about the breakpoint types and usage, see Breakpoint types.

- PROBLEM

  The **PROBLEM** panel displays compilation problems of apps. You can click a record to go to the corresponding line in the file.



2.18.6.3. Breakpoint types

App Studio supports normal line breakpoints, method breakpoints, and exception breakpoints.

Normal line breakpoint

You can click the blank section next to a line in the current file to generate a breakpoint for that line. The breakpoint also appears on the Breakpoint panel.

Method breakpoint

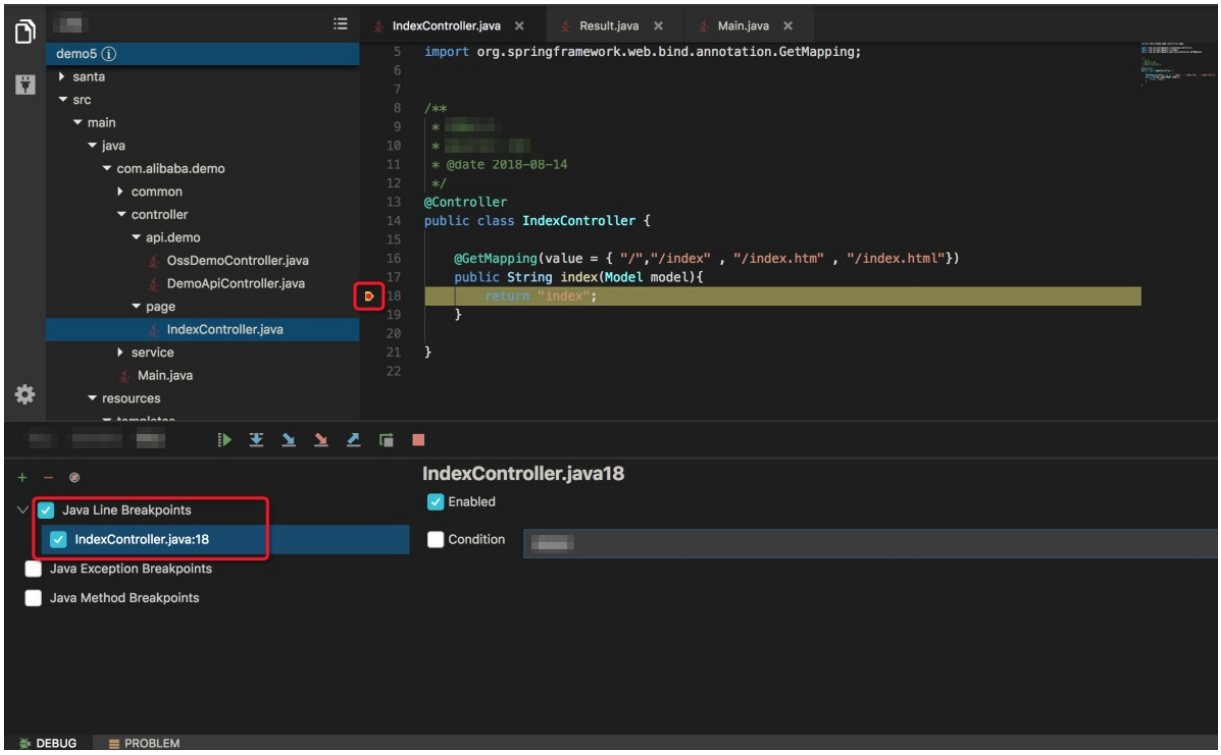
Different from a line breakpoint or an exception breakpoint, a method breakpoint triggers two events, namely, entry and exit. You can manually add a method breakpoint, or set a breakpoint at the place where the method is defined.

If the method breakpoint is triggered, the program stops when stepping into or out of the method.

Exception breakpoint

If an exception breakpoint is set, the program stops when encountering the exception.

As shown in the following figure, after index is triggered, the program stops in line 23 because **NullPointerException** appears.



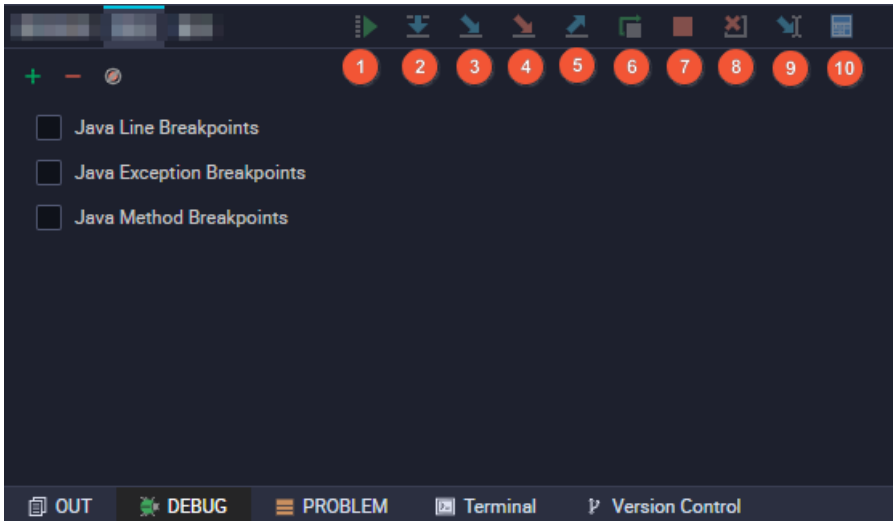
2.18.6.4. Breakpoint operations

The Breakpoint panel displays the breakpoints that are currently set. This topic describes how to operate breakpoints.

Breakpoints can be classified into normal line breakpoints, method breakpoints, and exception breakpoints. For more information, see Breakpoint types.

Debugging buttons

You can perform the debugging operations by clicking the following buttons listed in the table:



No.	Feature	Description
1	**Continue**	Resumes the current breakpoint to continue the current thread.
2	**Step Over**	Runs to the next line.
3	**Step Into**	Steps into a method.
4	**Force Step Into**	Forcibly steps into a method of a class not to be stepped into. Different from **Step Into**, **Force Step Into** enables you to step into a method from a built-in Java library.
5	**Step Out**	Steps out of the current method.
6	**Restart**	Currently, the **Restart** button is not perfect enough and may not be able to clean up the program. This button is being optimized.
7	**Stop**	Stops debugging.
8	**Drop Frame**	Deletes the current stack and returns to the previous method.
9	**Run to Cursor**	Runs to the current line of code. You can set a temporary breakpoint in a line.
10	**Evaluate Expression**	Calculates an expression.

2.18.6.5. Terminal

You can start multiple terminals in App Studio.

The **Terminal** tab appears in the lower part of the page.

App Studio supports common shell commands such as **ls** and **cat** and interactive commands such as **vi** and **top**.

2.18.6.6. Hot code replacement

Using the hot code replacement feature, you can edit the running code of an app and make the edits effective without restarting the app.

For example, after you edit the code while debugging a Spring Boot-based app, you do not need to restart the app. The edited code takes effect once it is saved. App Studio supports this feature by default.

App Studio also supports hot code replacement while an app is running. To trigger hot code replacement, you only need to save the file without installing any plug-in or manually compiling the file.

If you are editing the code in Debug mode, App Studio automatically deletes the current running stack and returns to the method entry.

Configure hot code replacement in Run mode

Enable hot code replacement on the Run/Debug Configurations page.

After you click Run or Debug, the output information of the HotCode2 plug-in appears on the OUT tab.

Save the file after editing it.

Configure hot code replacement in Debug mode

You can use the native Java Debug Interface (JDI) to enable hot code replacement in Debug mode. However, due to Java Virtual Machine (JVM) restrictions, hot code replacement is unavailable when a method is added to or deleted from a class. You can save the file to trigger hot code replacement.

> **Note** The native JVM supports hot code replacement for operations such as adding or deleting a class. However, hot code replacement is unavailable when you change the class structure.

2.18.7. WYSIWYG designer

2.18.7.1. Get started with the WYSIWYG designer

This topic describes basic operations in the WYSIWYG designer, including creating a project and building a visual page.

Create a project

1. Log on to the DataWorks console.
2. On the DataStudio page that appears, click the DataWorks icon in the upper-left corner and choose **All Products > App Studio**. The **Projects** page appears.
3. Click **Projects** in the left-side navigation pane. On the page that appears, click **Create Project from Code**.
4. On the **Create Project** page, set the **Name** and **Description** parameters, and set **Select the runtime environment** to **appstudio**.
5. After the configuration is completed, click **Submit**.

Build a visual page

Open a project created by using the WYSIWYG designer. Go to the *santa/pages* directory in your project.

Double-click a .santa file to go to the WYSIWYG designer. For example, you can double-click the file named *home.santa*.

You can also right-click **pages** and choose **Create > Template** to develop the page based on a template.

The WYSIWYG designer consists of the component menu and operation panel.

- Component menu

The component menu lists all components that the WYSIWYG designer presets, including **layout components**, **basic components**, **form components**, **chart components**, and **advanced components**.

Select a component from the component menu and drag and drop it to the visual operation section. Click the component. The **Component Settings** panel appears on the right.

On the **Component Settings** panel, you can configure the component on the **Properties**, **Style**, and **Advance** tabs.

- Operation panel

  You can click the corresponding icon on this panel to **undo an operation**, **redo an operation**, **preview the rendering result**, **enable the code mode**, **use the global style**, **configure the navigation**, **configure a global data flow**, **deploy as a template**, and **save edits**.

  Click the **Configure Navigation** icon in the upper-right corner to go to the navigation configuration page. For more information, see Navigation configuration.

Configure a global data flow

For more information about how to configure a global data flow, see Global data flow.

On the Component Settings panel, you can configure the component on the **Properties**, **Style**, and **Advance** tabs.

- Configure component properties

  On the Properties tab, you can visually configure component properties.

  Based on the rules for configuring component properties, a visual form is generated on the Properties tab. After you configure component properties in this form, the WYSIWYG designer re-renders the component in the visual operation section based on the new properties. You can view the rendering results of the component with different properties in real time.

- Configure component styles

  On the Style tab, you can configure the styles of a component.

  A visual panel for configuring common styles is provided on the Style tab. On this panel, you can customize the basic styles of a component, including the layout, text, background, border, and effect.

  After you add or modify the component styles on this tab, the WYSIWYG designer collects all the style settings and re-renders the component in the visual operation section based on the new component style. You can view the component configuration effect in real time.

- Configure association between components

  On the Advanced Settings tab, you can configure association between components.

  Select a component in the visual operation section and click the **Advance** tab. The properties of the selected component are listed on the left of the tab. Click the Magnifier icon on the right and select the component to be associated to your selected component.

  The properties of the associated component appear on the right of the tab.

  Select a property, for example, searchParams, in the left property list and connect it to a property, for example, requestParams, in the right property list.

In this way, any change of the searchParams parameter of the left component is transferred to the requestParams parameter of the right component in real time. This achieves property-based association between the two components.

Configure the code mode

By using the code mode, you can implement complex interactions in a more advanced way. For more information, see Code mode.

Save, preview, run, and hot code replacement

For more information, see Save, preview, run, and hot code replacement.

2.18.7.2. Code mode

By using the code mode, you can implement complex interactions in a more advanced way.

Click the **Code Mode** icon in the upper-right corner of the operation panel to enable the code mode.

The WYSIWYG designer uses domain-specific language (DSL) at the intermediate layer to switch between the visualization mode and code mode. DSL can be considered as a simplified version of React. The DSL syntax is basically the same as the React syntax.

As shown in the code section in the preceding figure, DSL uses a tag to describe a component. The tag properties are the component properties. The property value can be of a simple data type such as a string or a number. The property value can also be an expression. You can enter `state.xxx` to obtain data from the global data flow.

The code mode has the following features:

- If you drag and drop a component or configure the component properties in the visualization section, the edits are updated in the code in real time.
- If you edit the code in the code section, the edits are updated in the visualization section in real time.
- The drag-and-drop operation and component property configuration in the visualization section and code edits in the code section can be converted between each other.

2.18.7.3. DSL syntax

Domain-specific language (DSL) is a component-based language developed based on the features of React JSX and Vue templates and is more suitable for UI layout design.

JSX

The DSL syntax is similar to the JSX syntax in the React.render method. The following section provides a brief description of JSX:

- You can use `{}` to switch an HTML scope to a JavaScript scope. In a JavaScript scope, you can write any valid JavaScript expression. The return value appears on the page, for example, `<div>{'Hello' + ' Relim'}</div>`.

> **Note** You can write any JavaScript expressions such as computing statements or literals in `{}`.

- An HTML tag is used to switch a JavaScript scope to an HTML scope, for example, `{<div>Hello Relim</div>}`.
- The HTML scope and JavaScript scope can be nested, for example, `{<div>{'Hello' + ' Relim'}</div>}`.

Valid JavaScript expressions

```
// Computing statements
{aaa} // √ Variable aaa must be defined.
{aaa * 111} // √
{1 == 1 ? 1 : 0} // √
{/^123/.test(aa)} // √
{[1,2,3].join('')} // √
{(()=>{return 1})()} // The self-executing function. √

// Literals
{1}
{true}
{[11,22,33]} // √
{{aa:"11",bb:"22"}} // √
{()=>1} // Describe a function, which is valid but meaningless. √
```

> **Note** If certain complex logic must be implemented by multiple computing statements rather than only one statement, you can wrap the logic in a self-executing function, which must be a valid expression. The following statements provide an example:

```
{(function(){
    // Sum the even digits of a number array.
    var input = [1,2,3,4,5,6,7,8,9,10];
    var temp = input.filter(i => i % 2 == 0)
    return temp.reduce((buf, cur) => buf + cur, 0)
})()}
```

Invalid JavaScript expressions

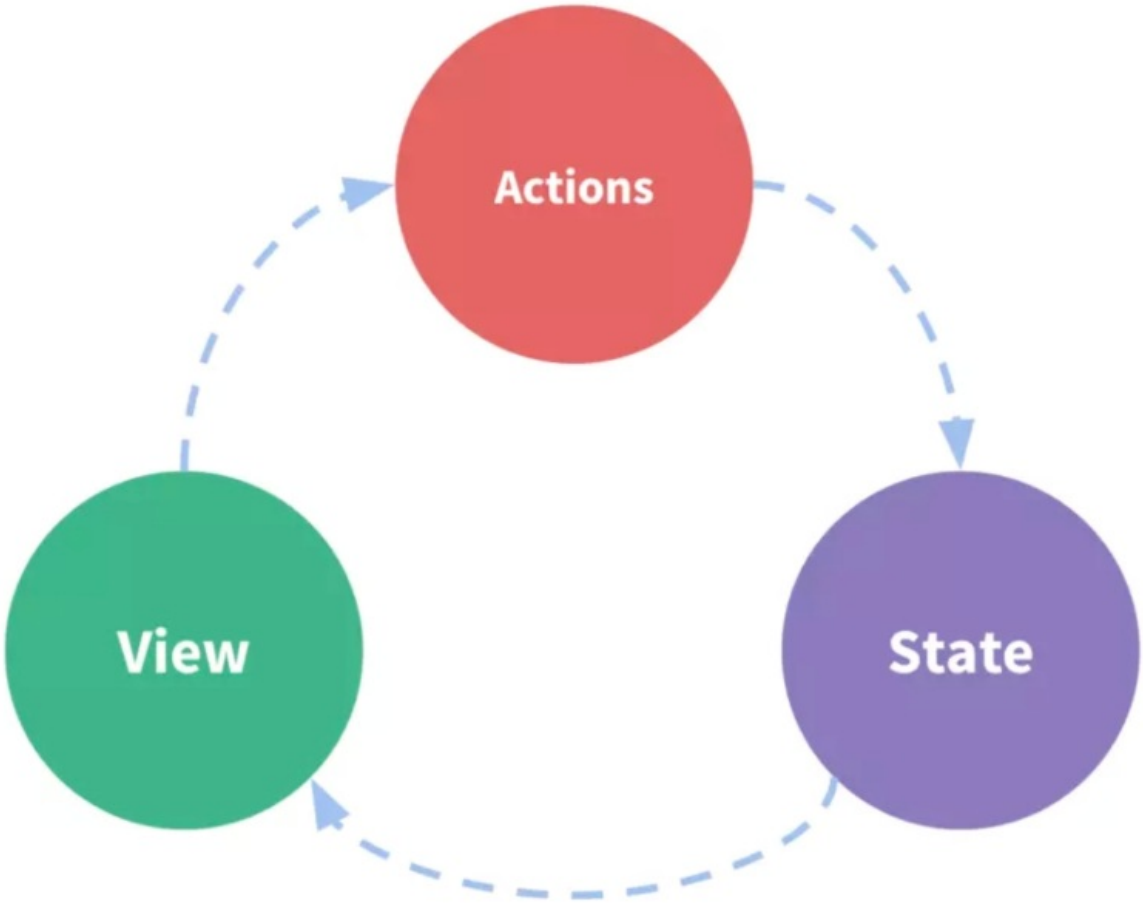
```
{ var a = 1 } // The value assignment statement.
{ aaa * 111; 2} // Multiple statements separated with semicolons (;).
```

2.18.7.4. Global data flow

A global data flow is used for front-end data management. For multiple components that need to share a state, it is difficult to transfer the state among them. To resolve this issue, you can extract the shared state and use a global data flow to transfer it to all related components.

Principles

In a global data flow, global data is transferred in a globally unique way. Once the data declared in global data changes, the data flow shown in the following figure is executed.



1. A component triggers an action when, for example, a user clicks the component.
2. The action triggers global data changes.
3. Upon the global data changes, components that reference the global state are automatically re-rendered.

Scenarios

A global data flow is applicable to the association of two or more components on a page. You can refine public data into global data for unified management, and then use a global data flow to associate two or more components.

Configure a global data flow

1. Click the **Global Data Flow Settings** icon in the upper-right corner of the operation panel.

2. In the **Global Data Flow Settings** dialog box that appears, set **Variable Name** and **Value**.
   - The variable value can be a number, character string, or JSON string.
   - If the variable value is declared as an API endpoint, data obtained from the API is automatically used as the value of the variable name.
3. Click **Save**.

Use a global data flow

- Obtain global data

  Use `state.name` in the component to obtain global data.

  ```
  <Input value={state.name} />
  ```

- Modify global data

  Use the $setState() method in the component to modify global data.

  ```
  <Input onChange={value => $setState({ name: value })} />
  ```

  Note You must use the $setState() method to modify global data. If you use `state.name = 'new value'`, re-rendering cannot be triggered.

2.18.7.5. Save, preview, run, and hot code replacement

In the WYSIWYG designer, you can perform operations such as saving edits, previewing the rendering result, running an app, or making edits in hot code replacement mode.

Save edits

The WYSIWYG designer periodically saves your edits. You can also click the **Save** icon in the upper-right corner of the operation panel to save edits.

Preview the rendering results

In the WYSIWYG designer, code in the operation section is in the editable status. However, special processing is added for the editable status of some components. For these components, you can run the rendering logic only when the app is running. To preview the rendering result, click the Preview icon in the upper-right corner of the operation panel.

Run an app

In the WYSIWYG designer, you can open and edit only one santa file at a time. To view the effect of the entire app,

click the Run Program icon on the Debug panel of App Studio to run the app.

Make edits in hot code replacement mode

If you are not satisfied with any page after running the app, you can edit the code in the WYSIWYG designer and save the edits.

The edited code takes effect on the running page in hot code replacement mode.

2.18.7.6. Navigation configuration

This topic describes how to configure the site navigation in the WYSIWYG designer.

The WYSIWYG designer provides each app with a public page header, a public bottom bar, and public sidebars, where you can configure various menus and themes. You can also specify whether to display the public header, bottom bar, and sidebars as required.

Click the **Navigation Settings** icon in the upper-right corner of the operation panel to go to the page for configuring the navigation of an app.

Configure the public header

You can configure the public header based on your business requirements.

Parameter	Description
Enabled	Specifies whether to display the public header.
Theme	The theme of the public header. You can select a dark or light theme.
Logo Image	The logo image of the site. You can enter an image URL or upload a local image.
Title	The title of the site.
Fix to Page Top	Specifies whether to fix the public header to the top of the page. If you turn on this switch, the public header stays at the top of the page when the page scrolls.
Menu Items	The menu items such as the link name and link URL that are displayed in the public header.

Configure the sidebars

You can configure the sidebars based on your business requirements.

Parameter	Description
Enabled	Specifies whether to display the sidebars.
Theme	The theme of the sidebars. You can select a dark or light theme.
Enable Folding	Specifies whether the sidebar menus can be hidden.

2.19. Workspace management

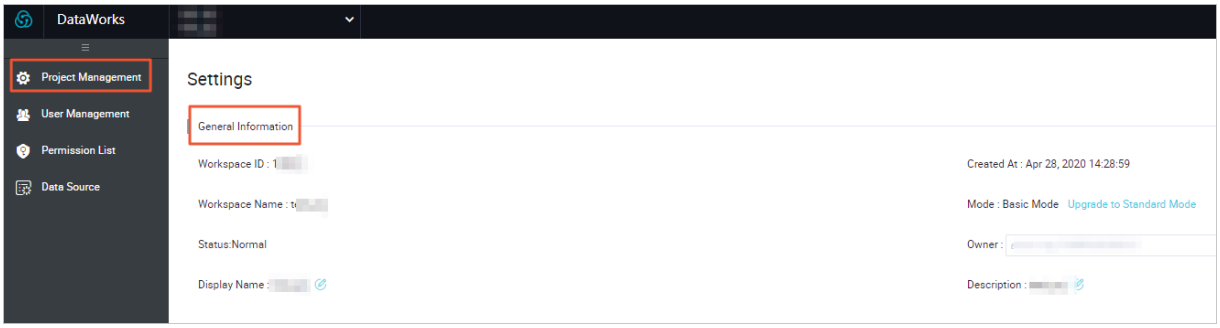
2.19.1. Configure a workspace

On the Project Management page of a workspace, you can manage and configure the workspace.

Go to the Project Management page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click in the upper-right corner.
3. On the **Project Management** page, configure the workspace as required in the **Basic properties**, **Scheduling properties**, **Security Settings**, and **Compute Engine information** sections.

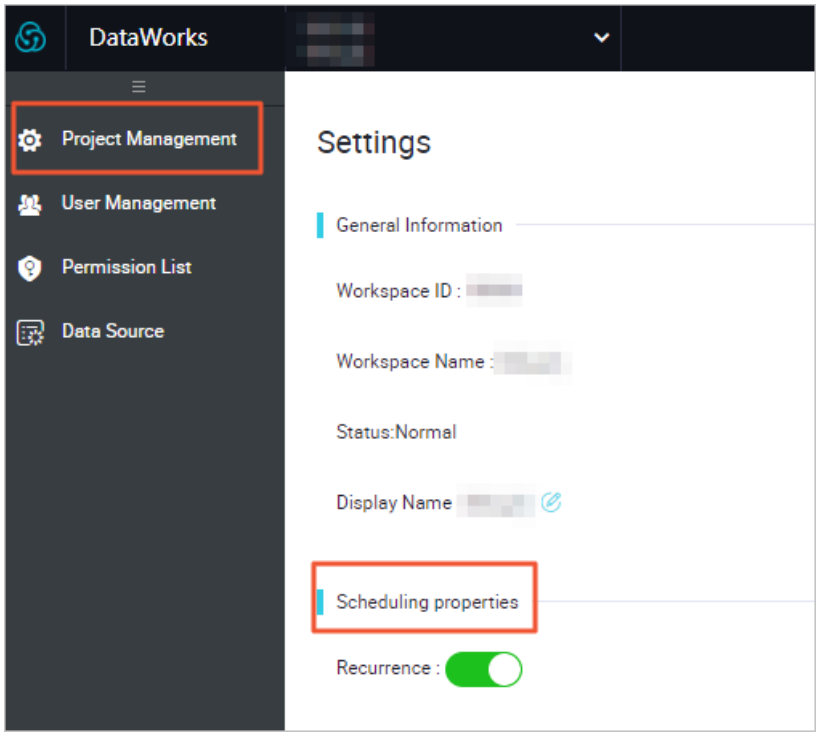
Basic properties section



Parameter	Description
Workspace ID	The ID of the workspace.
Workspace name	The name of the workspace. The name must start with a letter and can contain only letters and digits. It is not case-sensitive. It uniquely identifies the workspace and cannot be changed after the workspace is created.
Status	The status of the workspace.
Display name	The display name that is used to identify the workspace. It can contain only letters and digits. You can change the display name as required.
Creation date	The time when the workspace was created, which cannot be changed.
Mode	The mode of the workspace. Valid values: **Basic Mode** and **Standard Mode**.
The head of	The owner of the workspace. The owner of the workspace cannot be changed.
Description	The description of the workspace, which provides comments on the workspace. You can modify the description as required. The description can be up to 128 characters in length and can contain letters, special characters, and digits.

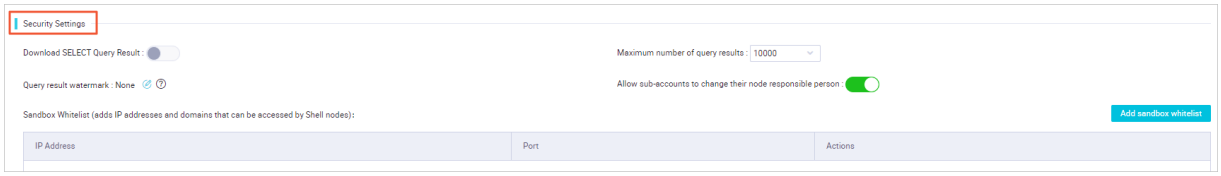
Scheduling properties section

In the **Scheduling properties** section, you can specify whether to enable the recurrence feature for the current workspace.



Recurring nodes can be run in this workspace only after you turn on the **Enable scheduling cycle** switch.

Security Settings section



Parameter	Description
Allow download of select results	Specifies whether to allow workspace members to download data query results in DataStudio. If you turn on this switch, workspace members can download data query results returned by SELECT statements in DataStudio.
Maximum number of query results	The maximum number of data records that can be returned for each query. Valid values: **10**, **100**, **500**, **1000**, **5000**, and **10000**. Default value: **10000**. Assume that you set the **Maximum number of query results** parameter to **1000**. Go to the **DataStudio** page 5 minutes later. In the left-side navigation pane, click **Ad-Hoc Query**. On the Ad-Hoc Query tab, create an ODPS SQL node and run the node to query data in a table that has more than 1,000 data records. 1,000 records are returned.
Query result watermark	Specifies whether to display watermarks for the query results.

Parameter	Description
Allow sub-accounts to change their own node owners	Specifies whether to allow RAM users to change the owner for their nodes.
Sandbox whitelist (configure IP addresses or domain names that shell tasks can access)	The whitelist of IP addresses and domain names that a Shell node run on the default resource group can access.

To add an IP address to the whitelist, perform the following steps:

1. In the **Security Settings** section, click **Add sandbox whitelist**.
2. In the **Add sandbox whitelist** dialog box, enter the IP address and port number in the **Address** and **Port** fields respectively.
3. Click **Confirm**.

Computing Engine information section

DataWorks supports the following types of compute engines: MaxCompute, E-MapReduce, Realtime Compute, and Graph Compute.

MaxCompute

1. Click the **MaxCompute** tab in the **Compute Engine information** section. On this tab, you can view the information about all available MaxCompute instances in the current workspace.
2. Click **Add instances**.
3. In the **Add MaxCompute engine** dialog box, set the **MaxCompute project name** and **MaxCompute access identity** parameters.
4. Click **Confirm**.

E-MapReduce

1. Click the **E-MapReduce** tab in the **Computing Engine information** section. On this tab, you can view the information about all available E-MapReduce clusters in the current workspace.
2. Click **Add instances**.
3. In the **New EMR cluster** dialog box, set relevant parameters.

Parameter	Description
Instance display name	The display name of the E-MapReduce cluster to be added.
Region	The region where the current workspace resides.
Access ID	The AccessKey ID of the account that is authorized to access the E-MapReduce cluster.
Access Key	The AccessKey secret of the account that is authorized to access the E-MapReduce cluster.
Cluster ID	The ID of the E-MapReduce cluster.

Parameter	Description
EmrUserID	The ID of the user who created the E-MapReduce cluster.
Project ID	The ID of the project in the E-MapReduce cluster.
YARN resource queue	The name of the resource queue in the E-MapReduce cluster. Unless otherwise specified, set the value to *default*.
Endpoint	The endpoint of the E-MapReduce cluster. You can obtain the endpoint from the E-MapReduce console.

4. Click **Confirm**.

Realtime Compute

1. Click the **Real-time computing** tab in the **Computing Engine information** section. On this tab, you can view the information about all available Realtime Compute projects in the current workspace.
2. Click **Add engine service**.
3. In the **New Blink engine** dialog box, set the **Enter the Realtime Compute project name to add** parameter.

> **Notice** A Realtime Compute project can be bound to only one DataWorks workspace. After a Realtime Compute project is bound to a DataWorks workspace, the Realtime Compute project cannot be used in other DataWorks workspaces. After you bind a Realtime Compute project to a workspace, you cannot unbind it. Use caution when you bind a Realtime Compute project.

4. Click **Binding**.

Graph Compute

1. Click the **GraphCompute** tab in the **Computing Engine information** section.
2. Click **Bind a GraphCompute instance**.

> **Notice** A Graph Compute instance can be bound to only one DataWorks workspace. After a Graph Compute instance is bound to a DataWorks workspace, the instance cannot be used in other DataWorks workspaces.

3. In the **Bind a GraphCompute instance** dialog box, set relevant parameters.

Parameter	Description
Instance display name	The display name of the Graph Compute instance to be bound to the workspace.
GraphCompute instance name	The name of the Graph Compute instance.

4. Click **Binding**.

2.19.2. Manage workspace members

This topic describes how to add and delete members and change the roles of members on the User Management page.

Go to the User Management page

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click in the upper-right corner.
3. In the left-side navigation pane, click **User Management**.You can view the information about each member in the following columns on the **User Management** page:
   - **Members**: the display name of the Apsara Stack tenant account or RAM user that the member uses to log on to DataWorks.
   - **Cloud account**: the Apsara Stack tenant account or RAM user that the member uses to log on to DataWorks.
   - **Role**: the roles of the member in the current DataWorks workspace. A member can have one or more of the following roles: workspace owner, workspace administrator, developer, administration expert, deployment expert, visitor, and security expert.

     > **Note** The workspace administrator can assign roles to RAM users as required. You can click **Permission List** in the left-side navigation pane to view the specific permissions of different member roles.

   - **Join TIME**: the time when the member was added to the workspace.
   - **Operation**: the operation that you can perform on the member. You can delete all members except the workspace owner from the workspace.

Add members

1. On the **User Management** page, click **Add Member** in the upper-right corner.
2. In the **Add Member** dialog box, select one or more accounts to add in the **Account to be added** list.
3. Click **>** to move the selected accounts to the **Added account** list.
4. Select the roles next to **Batch role setting**. You can select one or more of the following roles: **Space Administrator**, **Development**, **Operation & Maintenance (O & M)**, **Deployment**, **Visitors**, and **Security Administrator**.
5. Click **Confirm**.

Remove a member

1. On the **User Management** page, find the target member and click **Delete** in the Operation column.
2. In the **Member remove this workspace** message, click **OK**.

Assign a role to a member

Click the drop-down list in the **Role** column of the member and select the role to be assigned.

Revoke a role from a member

Click x next to the role in the Role column.

2.19.3. Permission list

DataWorks provides seven roles: workspace owner, workspace administrator, developer, administration expert, deployment expert, visitor, and security expert. You cannot grant the role of the workspace owner to other workspace members. This topic describes the permissions of these roles. In the following tables, Yes indicates that a role has the corresponding permission, and No indicates that a role does not have the corresponding permission.

Data management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Delete self-created tables	Yes	Yes	Yes	No	No	No	No
Specify categories for self-created tables	Yes	Yes	Yes	No	No	No	No
View favorite tables	Yes	Yes	Yes	No	No	No	No
Create tables	Yes	Yes	Yes	No	No	No	No
Unhide self-created tables	Yes	Yes	Yes	No	No	No	No
Modify the schemas of self-created tables	Yes	Yes	Yes	No	No	No	No
View self-created tables	Yes	Yes	Yes	No	No	No	No
View the content of self-submitted permission requests	Yes	Yes	Yes	No	No	No	No
Hide self-created tables	Yes	Yes	Yes	No	No	No	No
Specify the time-to-live (TTL) for self-created tables	Yes	Yes	Yes	No	No	No	No
Request permissions on tables created by others	Yes	Yes	Yes	No	No	No	No
Delete tables	No	Yes	Yes	No	No	No	No
Update tables	No	Yes	Yes	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Preview data	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Preview table data of other organizations	Yes	Yes	No	No	No	No	No

Deployment management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Create deployment tasks	Yes	Yes	Yes	Yes	No	No	No
View the list of deployment tasks	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete deployment tasks	Yes	Yes	Yes	Yes	No	No	No
Run deployment tasks	Yes	Yes	No	Yes	Yes	No	No
View the content of deployment tasks	Yes	Yes	Yes	Yes	Yes	Yes	No

Buttons

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Button: Stop	Yes	Yes	Yes	No	No	No	No
Button: Format	Yes	Yes	Yes	No	No	No	No
Button: Edit	Yes	Yes	Yes	No	No	No	No
Button: Run	Yes	Yes	Yes	No	No	No	No
Button: Zoom In	Yes	Yes	Yes	No	No	No	No
Button: Save	Yes	Yes	Yes	No	No	No	No
Button: Show/Hide	Yes	Yes	Yes	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Button: Delete	Yes	Yes	Yes	No	No	No	No

Code development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Save and commit nodes	Yes	Yes	Yes	No	No	No	No
View the code of nodes	Yes	Yes	Yes	Yes	Yes	Yes	No
Create nodes	Yes	Yes	Yes	No	No	No	No
Delete nodes	Yes	Yes	Yes	No	No	No	No
View the node list	Yes	Yes	Yes	Yes	Yes	Yes	No
Run nodes	Yes	Yes	Yes	No	No	No	No
Edit the code of nodes	Yes	Yes	Yes	No	No	No	No
Download files	Yes	Yes	Yes	No	No	No	No

Function development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View function details	Yes	Yes	Yes	Yes	Yes	Yes	No
Create functions	Yes	Yes	Yes	No	No	No	No
Query functions	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete functions	Yes	Yes	Yes	No	No	No	No

Node types

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Node type: Machine Learning	Yes	Yes	Yes	No	No	No	No
Node type: ODPS MR	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Data Sync	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: ODPS SQL	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: XLIB	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Shell	Yes	Yes	Yes	Yes	Yes	Yes	No
Node type: Zero-Load Node	Yes	Yes	Yes	Yes	Yes	Yes	No

Resource management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the resource list	Yes	Yes	Yes	Yes	Yes	Yes	No
Delete resources	Yes	Yes	Yes	No	No	No	No
Create resources	Yes	Yes	Yes	No	No	No	No
Upload Python files	Yes	Yes	Yes	No	No	No	No
Upload JAR files	Yes	Yes	Yes	No	No	No	No
Upload TXT files	Yes	Yes	Yes	No	No	No	No
Upload files as Archive resources	Yes	Yes	Yes	No	No	No	No

Workflow development

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Run or stop workflows	Yes	Yes	Yes	No	No	No	No
Save workflows	Yes	Yes	Yes	No	No	No	No
View workflows	Yes	Yes	Yes	Yes	Yes	Yes	No
Commit the code of nodes	Yes	Yes	Yes	No	No	No	No
Modify workflows	Yes	Yes	Yes	No	No	No	No
View the workflow list	Yes	Yes	Yes	Yes	Yes	Yes	No
Change the workflow owner	Yes	Yes	No	No	No	No	No
View the code of nodes	Yes	Yes	Yes	No	No	No	No
Delete workflows	Yes	Yes	Yes	No	No	No	No
Create workflows	Yes	Yes	Yes	No	No	No	No
Migrate database tables	Yes	Yes	Yes	No	No	No	No
Create folders	No	Yes	Yes	No	No	No	No
Delete folders	No	Yes	Yes	No	No	No	No
Modify folders	No	Yes	Yes	No	No	No	No
Export workflows	No	Yes	Yes	Yes	No	No	No

Workspace management

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the basic information about a workspace	Yes	Yes	Yes	Yes	No	No	No
Create baselines	Yes	Yes	Yes	No	No	No	No
Delete baselines	Yes	Yes	Yes	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Edit baselines	Yes	Yes	No	No	No	No	No
Search for baselines	Yes	Yes	Yes	No	No	No	No
View baselines	Yes	Yes	No	No	No	No	No
Test connectivity	Yes	Yes	No	No	No	No	No
Create connections	Yes	Yes	No	No	No	No	No
Delete connections	Yes	Yes	No	No	No	No	No
Edit connections	Yes	Yes	No	No	No	No	No
Search for connections	Yes	Yes	No	No	No	No	No
View the connections configured for a workspace	Yes	Yes	Yes	Yes	Yes	No	No
Enable scheduling	Yes	Yes	No	No	No	No	No
View the settings of scheduling properties of nodes	Yes	Yes	No	No	No	No	No
Add workspace members	Yes	Yes	No	No	No	No	No
Change the roles of workspace members	Yes	Yes	No	No	No	No	No
View the members of a workspace	Yes	Yes	No	No	No	No	No
Remove workspace members	Yes	Yes	No	No	No	No	No
Search for workspace members	Yes	Yes	No	No	No	No	No
Modify the configurations of compute engines	Yes	Yes	No	No	No	No	No
View the configurations of compute engines	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Query the members of within the tenant	Yes	Yes	No	No	No	No	No
Modify the basic information about a workspace	Yes	Yes	No	No	No	No	No
View the security policies of compute engines	Yes	Yes	No	No	No	No	No
Modify the security policies of compute engines	Yes	Yes	No	No	No	No	No
Query the resource groups that are bound to a workspace	Yes	Yes	Yes	No	No	No	No
Query the servers in a resource group	No	Yes	No	No	No	No	No
Delete resource groups	No	Yes	No	No	No	No	No
Remove servers from a resource group	No	Yes	No	No	No	No	No
Configure resource groups for sync nodes	Yes	Yes	Yes	Yes	Yes	Yes	No
Bind multiple resource groups to a workspace	No	Yes	No	No	No	No	No
Add servers to a resource group	No	Yes	No	No	No	No	No
Create resource groups for a workspace	No	Yes	No	No	No	No	No
Query the projects to which a resource group is bound	No	Yes	No	No	No	No	No
Create connections	Yes	Yes	No	No	No	No	No
Edit connections	Yes	Yes	No	No	No	No	No
Share connections	Yes	Yes	No	No	No	No	No
Delete connections	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Initialize servers in a resource group	No	Yes	No	No	No	No	No
View the connections configured for a workspace	Yes	Yes	Yes	Yes	Yes	No	No
Test connectivity	Yes	Yes	No	No	No	No	No
Search for connections	Yes	Yes	No	No	No	No	No
Update the server status and slots of a resource group	No	Yes	No	No	No	No	No
Create real-time sync nodes	Yes	Yes	Yes	No	No	No	No

Workflow O&M

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the DAG	Yes	Yes	Yes	Yes	No	No	No
Go to the DataStudio page	Yes	Yes	Yes	No	No	No	No
View the DAG of an instance	Yes	Yes	Yes	Yes	No	No	No
View ancestor and descendant nodes in the DAG	Yes	Yes	Yes	Yes	No	No	No
View the list of workflows	Yes	Yes	Yes	Yes	No	No	No
View the operations logs of workflows	Yes	Yes	Yes	Yes	No	No	No
Perform smoke tests	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Generate retroactive data for nodes	Yes	Yes	Yes	Yes	No	No	No
Change the owner of a node	Yes	Yes	Yes	Yes	No	No	No
Unpublish workflows	Yes	Yes	Yes	Yes	No	No	No
View the details of workflows	Yes	Yes	Yes	Yes	No	No	No
View ancestor and descendant instances of an instance in the DAG	Yes	Yes	Yes	Yes	No	No	No
Pause instances	Yes	Yes	Yes	Yes	No	No	No
Restore instances	Yes	Yes	Yes	Yes	No	No	No
Terminate an instance	Yes	Yes	Yes	Yes	No	No	No
Terminate multiple instance at a time	Yes	Yes	No	Yes	No	No	No
View the list of instances	Yes	Yes	Yes	Yes	No	No	No
View operational logs	Yes	Yes	Yes	Yes	No	No	No
Rerun an instance	Yes	Yes	Yes	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	No	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	Yes	Yes	No	No	No
Search for instances	Yes	Yes	Yes	Yes	No	No	No
Set the status of an instance to Successful	Yes	Yes	Yes	Yes	No	No	No
View the lineage of nodes	Yes	Yes	Yes	Yes	Yes	No	No
View node details	Yes	Yes	Yes	Yes	Yes	No	No
View the operations logs of nodes	Yes	Yes	Yes	Yes	Yes	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Freeze and pause nodes	Yes	Yes	Yes	Yes	Yes	No	No
Unfreeze and resume nodes	Yes	Yes	Yes	Yes	Yes	No	No
Change the baseline for nodes	Yes	Yes	Yes	Yes	Yes	No	No
Resume instances	Yes	Yes	Yes	Yes	Yes	No	No
Delete instance dependencies	Yes	Yes	No	Yes	No	No	No
Change the running priority of instances	Yes	Yes	No	Yes	No	No	No
Forcibly rerun instances	Yes	Yes	No	Yes	No	No	No
View the lineage of instances	Yes	Yes	Yes	Yes	Yes	No	No
View instance details	Yes	Yes	Yes	Yes	Yes	No	No
View runtime logs of instances	Yes	Yes	Yes	Yes	Yes	No	No
View the baselines affected by instances	Yes	Yes	Yes	Yes	Yes	No	No
Unpublish nodes	Yes	Yes	No	Yes	No	No	No

Node maintenance

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Change the baseline for a node	Yes	Yes	Yes	Yes	No	No	No
Change the baseline for multiple nodes at a time	Yes	Yes	No	Yes	No	No	No
View the code of a node	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Change the owner of a node	Yes	Yes	Yes	Yes	No	No	No
Change the owner of multiple nodes at a time	Yes	Yes	No	Yes	No	No	No
Change the resource group for a node	Yes	Yes	Yes	Yes	No	No	No
Change the resource group for multiple nodes at a time	Yes	Yes	Yes	Yes	No	No	No
Perform smoke tests	Yes	Yes	Yes	Yes	No	No	No
Generate retroactive data for nodes	Yes	Yes	Yes	Yes	No	No	No
Delete instance dependencies	Yes	Yes	No	Yes	No	No	No
Pause instances	Yes	Yes	Yes	Yes	No	No	No
Resume instances	Yes	Yes	Yes	Yes	No	No	No
Refresh the attribute information about instances	Yes	Yes	Yes	Yes	No	No	No
Terminate an instance	Yes	Yes	Yes	Yes	No	No	No
Terminate multiple instance at a time	Yes	Yes	No	Yes	No	No	No
Change the running priority of instances	Yes	Yes	Yes	Yes	No	No	No
Refresh the dependencies of instances	Yes	Yes	Yes	Yes	No	No	No
Rerun an instance	Yes	Yes	Yes	Yes	No	No	No
Rerun multiple instances at a time	Yes	Yes	No	Yes	No	No	No
Set the status of an instance to Successful	Yes	Yes	Yes	Yes	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Create data quality rules	Yes	Yes	Yes	Yes	No	No	No
Delete data quality rules	Yes	Yes	Yes	Yes	No	No	No

Dashboard

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the number of baselines in the Overtime state	Yes	Yes	Yes	Yes	No	No	No
Remove a record from the dashboard	Yes	Yes	Yes	Yes	No	No	No
View the distribution of nodes by status	Yes	Yes	Yes	Yes	No	No	No
View the running information about nodes	Yes	Yes	Yes	Yes	No	No	No
View the distribution of nodes by type	Yes	Yes	Yes	Yes	No	No	No

Baseline checks

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View the metrics of baselines	Yes	Yes	Yes	Yes	No	No	No
View baselines	Yes	Yes	Yes	Yes	No	No	No

Monitoring and alerts

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
View notification messages	Yes	Yes	Yes	Yes	No	No	No
Disable an alert	Yes	Yes	Yes	Yes	No	No	No
Disable multiple alerts at a time	Yes	Yes	No	No	No	No	No
Enable or disable call notifications	Yes	Yes	Yes	Yes	No	No	No
Create custom notification rules	Yes	Yes	Yes	Yes	No	No	No
Delete custom notification rules	Yes	Yes	Yes	No	No	No	
Edit custom notification rules	Yes	Yes	Yes	Yes	No	No	No
View custom notification rules	Yes	Yes	Yes	Yes	No	No	No
View all events	Yes	Yes	Yes	Yes	No	No	No
View event details	Yes	Yes	Yes	Yes	No	No	No
View details of personal events	Yes	Yes	Yes	Yes	No	No	No

Data integration

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Resource consumption monitoring menu	Yes	Yes	No	No	No	No	No
View nodes	Yes	Yes	Yes	No	No	No	No
Edit nodes	Yes	Yes	Yes	No	No	No	No
Monitor resource consumption	Yes	Yes	No	No	No	No	No

Permission	Workspace owner	Workspace administrator	Developer	Administration expert	Deployment expert	Visitor	Security expert
Delete nodes	Yes	Yes	Yes	No	No	No	No
Migrate database tables	Yes	Yes	No	No	No	No	No

2.19.4. Manage connections

Connections are used to configure readers and writers during data integration. On the Data Source page of a workspace, you can view and add connections.

Procedure

1. Log on to the DataWorks console.
2. On the **DataStudio** page, click  in the upper-right corner.
3. In the left-side navigation pane, click **Data Source**. On the **Data Source** page, you can filter connections by conditions such as **Connect To** and **Connection Name**.

   Click **Add Connection** in the upper-right corner to add a connection. For more information, see Data sources.

3.Realtime Compute

3.1. What is Realtime Compute?

Realtime Compute is a big data processing platform that analyzes streaming data in real time based on Apsara Stack. You can create streaming data analysis and computing jobs by using Alibaba Cloud Flink SQL. When you use Flink SQL, you do not need to develop the underlying logic for streaming data processing.

As the demands for high data timeliness and operability increase, software systems need to process more data in less time. In traditional models for big data processing, online transaction processing (OLTP) and offline data analysis are separately performed at different times.

Realtime Compute is designed to respond to the increasing demand for the timeliness of data processing. The business value of data decreases as time passes by. Therefore, data must be computed and processed as soon as possible after it is generated. Traditional models for big data processing follow the scheduled processing mode, which accumulates and processes data in a computing cycle that can last hours or even days. These models cannot satisfy the growing demand for real-time big data processing. Batch processing cannot meet the business requirements in scenarios in which an extremely low processing delay is required. These scenarios include real-time big data analysis, early warning and risk control, real-time forecasting, and financial transactions. Realtime Compute enables real-time processing over data streams. With Realtime Compute, you can shorten the data processing delay, easily implement real-time computational logic, and greatly reduce computing costs. This helps you meet business requirements for real-time processing of big data.

Streaming data

Big data can be viewed as a series of discrete events. These discrete events form event streams or data streams along a timeline. Streaming data is continuously generated from thousands of data sources and is typically sent in data records. Streaming data has a smaller scale than offline data. Each type of data is produced as a stream of events. Streaming data includes a wide variety of data, such as the log files generated by your mobile or web applications, online purchases, in-game player activities, information from social networks, financial trade centers, geospatial services, and telemetry data from connected devices in data centers.

Realtime Compute has the following advantages:

- Real-time and unbounded data streams

  Realtime Compute processes data streams in real time. Streaming data is continuously generated from data sources and is subscribed and consumed in chronological order. Data streams continuously flow into the Realtime Compute system. For example, when Realtime Compute processes data streams from website visit logs, the log data streams continuously enter the Realtime Compute system as long as the website is online. In Realtime Compute, unbounded data streams are processed in real time.

- Continuous and efficient computing

  Realtime Compute is an event-driven system in which unbounded event or data streams continuously trigger real-time computations. Each streaming data record triggers a computational task. Realtime Compute performs continuous and real-time computations over data streams.

- Real-time integration of streaming data

Realtime Compute writes the computing result of each streaming data record into the target data store in real time. For example, the system directly writes the data of a computed report to an ApsaraDB for RDS instance for report display. Realtime Compute continuously writes the results into the target data store in real time. Therefore, Realtime Compute can be viewed as a data source that generates data streams for the target data store.

3.2. Quick start

3.2.1. Log on to the Realtime Compute console

This topic describes how to log on to the Realtime Compute console.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, move the pointer over Products and click **Realtime Compute**.
5. Specify **Organization** and **Region**.
6. Click **Blink**.

3.2.2. Real-time security monitoring

3.2.2.1. Overview

With the wide application of digital technologies, every industry is facing the ever-increasing demand for data security, especially for real-time monitoring and alerting. To monitor streaming data and report alerts in real time, you need to ensure that the data is accurate and is processed instantly after it has been generated. To address these challenges, Realtime Compute allows you to perform JOIN operations on source tables of streaming data and dimension tables that include blacklists. The following sections describe a use case of real-time monitoring and alerting.

3.2.2.2. Preparations

Before proceeding with the development process in the Realtime Compute console, you must create a source table and a result table in the upstream and downstream data stores, respectively. You must also upload data to the source table.

Context

To simplify operations, this example organizes incoming streaming data based on a table: datahub_IpPlace.

datahub_IpPlace

Field name	Data type	Description
name	VARCHAR	The name
Place	VARCHAR	The place

The rds_dim dimension table is described as follows.

rds_dim

Field name	Data type	Description
name	VARCHAR	The name
Place	VARCHAR	The place

A result table named rds_IpPlace is obtained after a JOIN operation is performed on the datahub_IpPlace and rds_dim tables. This result table is described as follows.

rds_IpPlace

Field name	Data type	Description
name	VARCHAR	The name
Place	VARCHAR	The place

Create source and result tables

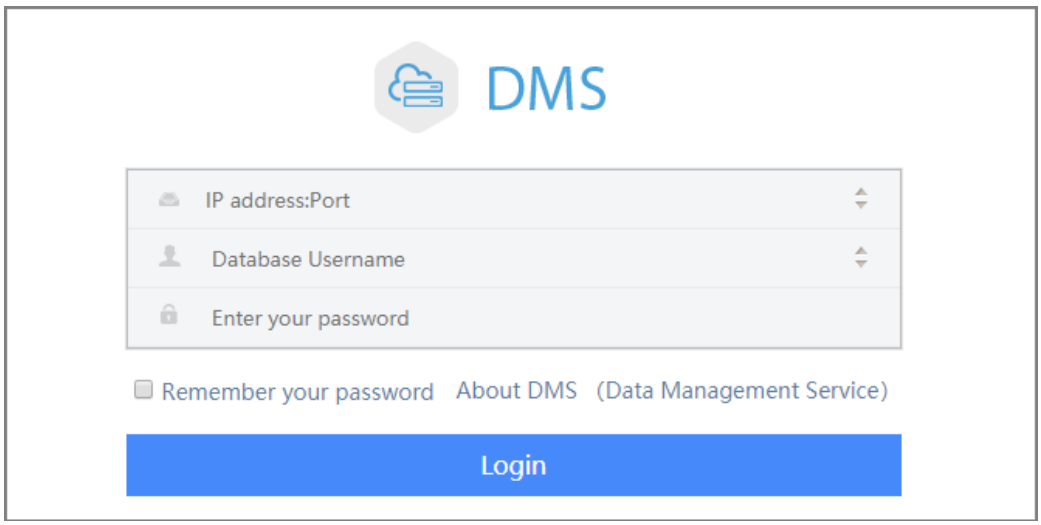
Procedure

1. Log on to the DataHub console. For more information, see the "Log on to the DataHub

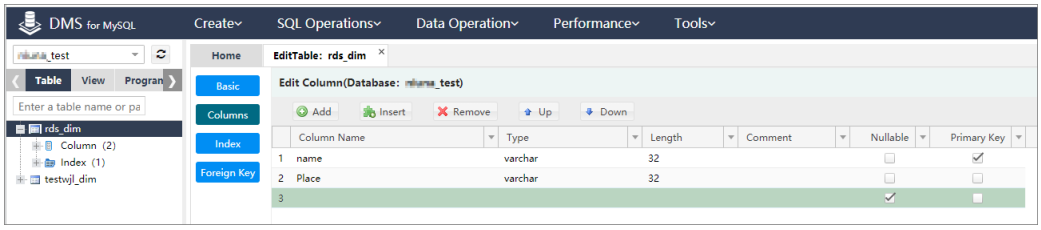
console" section in *DataHub User Guide*.

2. Create a project. For more information, see the "Create projects" section in *DataHub User Guide*.
3. On the page that shows the project details, create a topic. The following figure shows the schema of the source table. For more information, see the "Create a topic" section in *DataHub User Guide*.
4. Create an ApsaraDB for RDS instance. For more information, see the "Create an instance" section in *ApsaraDB for RDS User Guide*.
5. On the **Instances** page of the ApsaraDB for RDS console, click the target instance name.
6. On the **Basic Information** page, click **Log On to DB**.
7. Log on to the RDS database, and create a result table in the database.

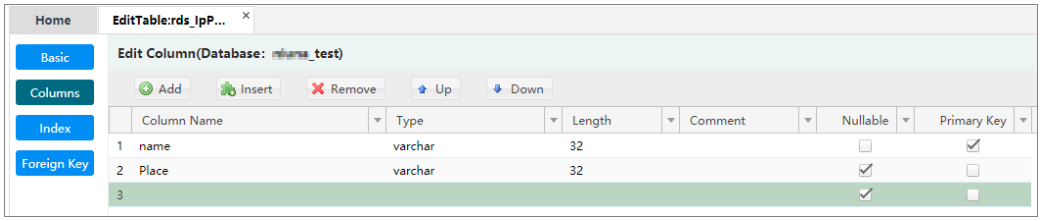
   Log on to the database

   Dimension table schema

   Result table schema

8. Upload data to DataHub.

Log on to the **DataHub** console. In the left-side navigation pane, click **Data Acquisition**, and then click **Upload File**. On the page that appears, double-click the project, click the target DataHub topic, and then click **Select File** to upload data. For more information, see the "Upload local files" section in *DataHub User Guide*.

9. Write data into the rds_dim dimension table.
   i. Log on to the RDS database. In the top navigation bar, choose **SQL Operations** > **SQL Window**.
   ii. On the **SQL Window** tab, enter the following code.

```
insert into 'rds_dim'('name','Place') VALUES('test01','beijing')
```

   iii. On the **SQL Window** tab, click **execute**.

3.2.2.3. Develop a job

After you create source and result tables in external data stores, you must register the data stores in the Realtime Compute console and create references to the source and result tables. After the data stores are registered, you can proceed with the development process.

Procedure

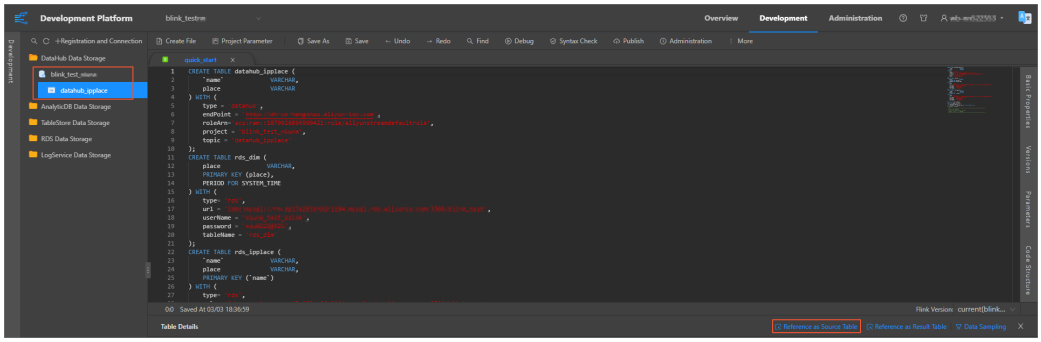
1. Log on to the Realtime Compute console to go to the homepage of Realtime Compute.
2. In the top navigation bar, click Development.
3. On the top of the page that appears, click the **Create File** icon.

Parameter description

Parameter	Description
File Name	The name of the file. The name must be 3 to 64 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter.
File Type	The type of the file. Valid values: FLINK_STREAM / SQL and FLINK_STREAM / DATASTREA.
Storage Path	The folder where the job SQL file is located. You can click the folder icon on the right side of an existing folder and create a subfolder.

4. Select DataHub as the data store for a source table. You can use the DataHub parameter settings and schema information that are automatically generated on the Data Storage tab. For more information, see Register a DataHub project.

   Use DataHub as the data store for a source table



5. Double-click a DataHub project in the **DataHub Data Storage** folder.
6. On the top of the section that appears, click **Reference as Source Table**.
7. Create a reference to the ApsaraDB for RDS dimension table. You must specify the required parameters and the table schema. Dimension tables cannot be registered on the Storage tab. The following sample code demonstrates how to create an ApsaraDB for RDS dimension table:

```
create table rds_input (
   place varchar,
   `name` varchar,
   primary key (place),
   period for system_time
) with (
   type = 'rds',
   url = 'yourURL',
   tableName = 'yourTableName',
   password = 'yourPassword'
);
```

8. Select RDS as the data store for the result table. You can use the RDS parameter settings and schema information that are automatically generated on the Data Storage tab. For more information, see Register an RDS instance.
9. To use RDS to store the result table, double-click the RDS Data Storage folder, double-click the target database, and then double-click the target table. On the top of the section that appears, click **Reference as Result Table**.
10. Edit SQL statements that implement the computing logic in the code editor.

```
INSERT INTO rds_output
SELECT
   t.`name`,
   w.place
FROM datahub_input1 as t
JOIN rds_intput FOR SYSTEM_TIME AS OF PROCTIME() as w
ON t.place = w.place
```

11. Click **Debug**.

12. Click **Publish**. After the computing logic is verified during the debugging, click **Publish** on the Development page to publish the job SQL file. Then, you can view the job on the Administration page of the Realtime Compute console and manage the job in the production environment, such as starting the job.

3.2.2.4. Administration

After you create and publish the job SQL file on the **Development** page, you can manage the job on the **Administration** page. For example, you can start, suspend, terminate, publish, or unpublish the job.

Procedure

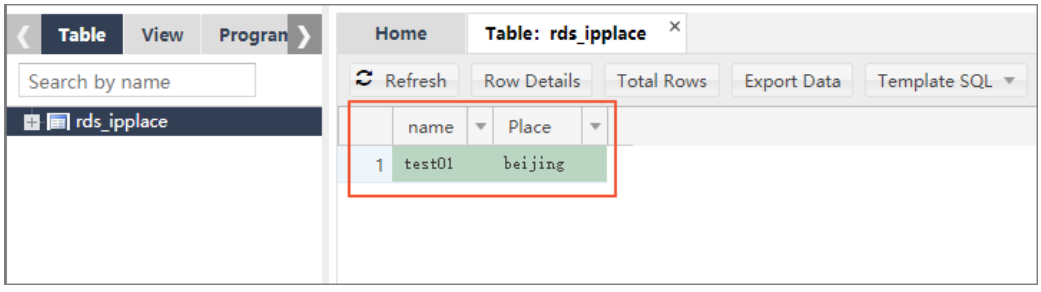
1. Go to the **Administration** page of the Realtime Compute console.
   i. Log on to the Realtime Compute console.
   ii. In the top navigation bar, click **Administration**.
2. Click **Start** for the `bj_dim_join` job.
3. Specify the **Start Time for Reading Data** parameter. The start time for reading data is also known as the start offset. The source data store remains connected to Realtime Compute for only a short period of time. Therefore, we recommend that you specify a time that is earlier than the current time.

   The Start Time for Reading Data parameter specifies the start time for reading data from the source data store.

Parameter	Description
Start Time for Reading Data	Specifies the start time for reading data from the source data store.
Enable Auto Upgrade	Specifies whether to enable auto upgrade.
Upgrade Time	Specifies the upgrade time range.
Offset	Specifies the range of data that is to be updated.

4. Click **OK** to start the job.
5. View the output data in the ApsaraDB for RDS data store.

   View the output data in the RDS data store



3.2.3. Frequently used words

3.2.3.1. Overview

Statistical analysis of frequently used words is widely applied in diverse fields, including the analysis of frequently used words in search engines, forums, and tags. For example, you can easily view the latest and most frequently searched words in microblogging websites through real-time statistics. Statistical analysis of frequently used words is, at its core, a simple word count job. In word count jobs for streaming data, real-time processing logic is used to analyze and display frequently used words in real time.

If you are new to working with big data computing, a word count job is for you to easily get started. The word count job in big data computing is similar to a `Hello, World!` program that is often the first program that a developer learns to write. The following topics take a word count job in Realtime Compute as an example to describe how to create a word count job based on real-time processing logic. This example helps you quickly get familiar with basic Flink SQL syntax and basic operations of Realtime Compute jobs, such as creating an SQL file for a job and publishing the job.

3.2.3.2. Code development

This topic uses a word count job as an example to describe how to create a Realtime Compute job.

Prerequisites

Before creating a word count job, a source table named stream_source and a result table named stream_result are created in external data stores. The stream_source table includes only one column. The column is named word and its data type is STRING. The stream_result table includes two columns. One column is named word and its data type is STRING. The other column is named cnt and its data type is BIGINT. The two tables are registered in Realtime Compute.

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. Right-click the folder where you want to store the job file.
4. Click **Create File**, and the **Create File** page appears.
5. On the page that appears, set the following parameters:
   - File Name: Set the value to `wordcount`.
   - File Type: Set the value to FLINK_STREAM/SQL.
   - Storage Path: Keep the default setting.
6. Enter the following code in the code editor.

> **Note** In the SQL statements for the word count job, the STRING data type for the referenced table must be declared as the VARCHAR data type.

```
 create table stream_source (word varchar);
create table stream_result (word varchar, cnt bigint);
insert into
   stream_result
select
   t.word,
   count (1)
from
   stream_source t
group by
   t.word;
```

The following section explains the SQL code.

Line 1 creates a reference to the stream_source source table.

> **Note** Streaming data continuously enters Realtime Compute and triggers stream processing procedures. Each streaming data record or each batch of data from the stream_source table triggers a stream processing procedure.

Line 2 creates a reference to the stream_result result table. The stream_result table stores the computing results of the word count job.

> **Note** Realtime Compute does not have built-in components for data storage, and the result data is stored in external data stores, such as ApsaraDB for RDS and Tablestore. This line of code creates a reference to a result table that contains the result data.

Lines 5 through 11 implement the computing logic: Realtime Compute reads data from the stream_source table and counts how often words occur based on incoming data records.

> **Note** Flink SQL supports most standard SQL statements. This allows you to easily and cost-effectively adopt Realtime Compute for stream processing.

The method of performing a word count job for stream processing is similar to that for batch processing. The word count job for stream processing continuously processes unbounded data streams until the job is terminated.

3.2.3.3. Code debugging

Realtime Compute provides a powerful debugging feature to verify SQL statements. You can debug Realtime Compute jobs by simulating data stores where streaming data, static data, and result data are stored.

> **Note**
>
> - To avoid negative impacts on online data stores, Realtime Compute is not allowed to read data from these data stores during the debugging process. Before debugging, you must prepare test data for input tables.
> - The outputs of INSERT operations are exported only to local screens. This does not affect online systems.

Debugging method

1. On the top of the **Development** page, click **Debug**.
2. On the page that appears, click **Download Template** and edit the template based on your debugging rules.

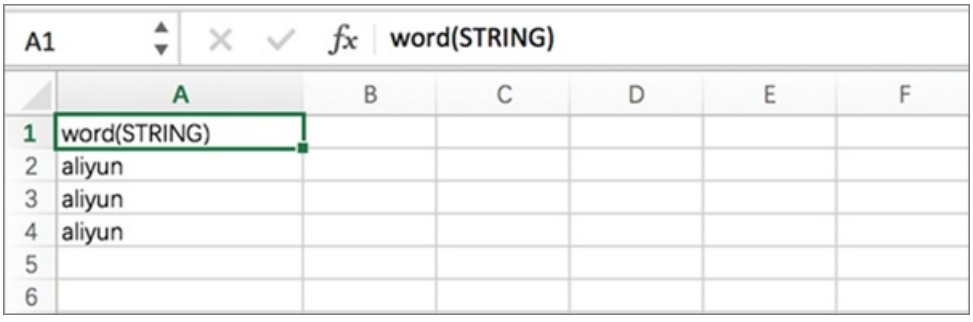
   > **Note** The file that is uploaded for debugging must meet the following requirements:
   >
   > - The file size cannot exceed 1 MB, and the file contains maximum of 1,000 records.
   > - The file must use UTF-8 encoding.
   > - Commas (,) cannot be used in test data, because the file uses the comma-separated values (CSV) format.
   > - Numeric values can be displayed only in the general format, and cannot be displayed in the scientific notation format.

3. Click **Upload** to upload the file.
4. Click **OK**.
5. View the debugging result in the output window.

Sample file for debugging the word count job

> **Note** The file for debugging is in the CSV format. We recommend that you use the following software applications to open and modify the template:
>
> - Excel for Windows users.
> - Vim or Sublime Text for MacOS users. We do not recommend that you use Number because it adds unnecessary fields when you modify CSV files.

Sample file for debugging the word count job



Test data

You can download test data and upload the data on the **Debug File** page.

> **Note** The *test data for statistical analysis of frequently used words* is unavailable for download in a PDF file. You can contact system administrators to download the test data.

View the debugging result

Real-time computing is triggered by data streams. Each data record from the stream_source table triggers a stream processing procedure. After each procedure is complete, a computing result is exported. The test file contains three data records. After each data record reaches Realtime Compute, a stream processing procedure is triggered. Therefore, a total of three data records are displayed on the screen. Realtime Compute uses the following computing logic:

- The first data record (aliyun) reaches Realtime Compute. This is the first time that the system has detected the word "aliyun." Therefore, the computing result is `<aliyun, 1>`, which is displayed on the screen.
- The second data record (aliyun) reaches Realtime Compute. The system detects an existing record of `<aliyun, 1>` and increases the value by one. Therefore, the computing result is `<aliyun, 2>`, which is displayed on the screen.
- The third data record (aliyun) reaches Realtime Compute. The system detects an existing record of `<aliyun, 2>` and increases the value by one. Therefore, the computing result is `<aliyun, 3>`, which is displayed on the screen.

The third computing result `<aliyun, 3>` is considered as the final output of the debugging. Another sample of test data is provided for you to test the debugging feature. You can use different samples of test data and view the debugging outputs.

3.2.3.4. Administration

After the SQL file has been verified, you can publish the SQL file for the job on the **Administration** page of Realtime Compute. Then, you can start the job. The job runs on a Realtime Compute cluster.

Procedure

1. On the **Development** page, click **Publish**. The **Publish New Version** dialog box appears.
2. In the **Resource Configuration** step, click **Next**.
3. In the **Check** step, click **Next**.
4. In the **Publish File** step, click **Publish**.
5. On the **Administration** page, view the published word count job.
6. Click **Start** in the **Actions** column of the word count job. The **Start** dialog box appears.
7. Specify **Start Time of Reading Data** and click **OK**. Then, the job runs on a Realtime Compute cluster.

Result

After the job is started, click the job name. The **Overview** page appears.

FAQ

Q: Why does the word count job have no input or output while it is running on the distributed compute clusters of Realtime Compute?

A: When you created the `my_source` and `my_result` tables, you did not specify the data storage type of the referenced data source. In this scenario, the source table is considered to be a random table of strings or digits, and the result table is considered to be discarded data.

3.2.4. Big screen service for the Tmall Double Eleven Global Shopping Festival

3.2.4.1. Overview

During Double 11, a big screen shows the total sales volume of Alibaba Group in real time. The big screen service is a highlight for the shopping festival.

Stream processing for the big screen service was previously based on Apache Storm that is an open source distributed real-time computation system. The Apache Storm-based development process took around one month. The application of Flink SQL reduces the period for the development of the big screen service to three days. The underlying layer of Realtime Compute removes the Apache Storm modules that are designed for execution optimization and troubleshooting. This achieves a higher processing efficiency for Realtime Compute jobs.

3.2.4.2. Scenario description

The streaming data input for the Tmall big screen service is the transaction data from the Tmall platform. The incoming transaction data is organized based on a two-dimensional table: `tmall_trade_detail`.

Field	Type	Description
tid	BIGINT	The order ID.
buyer_uid	BIGINT	The buyer ID.
seller_uid	BIGINT	The seller ID.
gmtdate	TIMESTAMP	The time when the order is completed.
payment	DOUBLE	The order amount.

Realtime Compute calculates two metrics based on the preceding table: the total number of orders and the total order amount up to the current time. The two metrics are written to an online RDS system and displayed on a big screen in real time. The online RDS system is used to store the result table: `tmall_trade_state`.

Field	Type	Description

Field	Type	Description
gmtdate	VARCHAR(16)	The date when the order is completed.
trade_count	BIGINT	The total number of orders.
trade_sum	DOUBLE	The total order amount.

The following topics describe how to build an end-to-end solution for the Tmall big screen service in around 10 minutes.

3.2.4.3. Preparations

Before editing Flink SQL statements for a Realtime Compute job, you must register data stores for source tables and result tables in Realtime Compute. This topic uses DataHub as an example to describe how to register a data store in Realtime Compute.

Create a DataHub topic

1. Log on to the DataHub console. For more information, see the "Log on to the DataHub console" section in *DataHub User Guide*.
2. If one or more projects have been created, click **View** in the **Actions** column for the target project. If no projects have been created, click **Create Project** to create a project and click **View** in the **Actions** column.
3. On the page that appears, click **Create Topic**.
4. Configure the topic based on the schema of the tmall_trade_state RDS table in the "Scenario description" section.

After performing these steps, you can edit Flink SQL statements for the Realtime Compute job.

Upload data to DataHub

You can upload data to the DataHub topic that you have created. To do this, follow these steps. Log on to the **DataHub** console. You can upload data to the DataHub topic that you have created. To do this, follow these steps:

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Data Acquisition**.
3. Click **Upload File**.
4. On the page that appears, double-click the target project and click the target DataHub topic.
5. Click **Select File** to select a file.
6. Click **Upload**.

To simplify the test procedure, you can use thetest data about the Double 11. You can download the data and then upload it to the DataHub topic for data collection.

3.2.4.4. Register a data store

The data store registration feature of Realtime Compute allows you to easily register DataHub topics, create tables, and reference data stores. To register a data store, follow these steps:

Procedure

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click the **Storage** tab.
4. Click the **DataHub Data Storage** folder.
5. On the top of the page, click **+Registration and Connection**.
6. Register a DataHub project in Realtime Compute. For more information about parameter settings, see Register a DataHub project.

   If you use ApsaraDB for RDS for MySQL to store the result data for data visualization, you must register an RDS data store in Realtime Compute. For more information, see Register an RDS instance.

3.2.4.5. Development

After the data has been collected to Realtime Compute, you can continue to edit Flink SQL statements.

1. Create a reference to the source.

   To create references to the DataHub source table and RDS result table, click Data Storage in the left-side navigation pane of the **Development** page in the Realtime Compute console, and perform the following operations:

   - Find the target DataHub topic, and click **Reference as Source Table**. Realtime Compute automatically parses the schema of the topic and adds the corresponding SQL statements to the **Development** page.
   - Find the target RDS table, and click **Reference as Result Table**. Realtime Compute automatically parses the schema of the table and adds the corresponding SQL statements to the **Development** page.

2. Edit Flink SQL statements.

   If you have created the DataHub topic and RDS table as described in the previous topics, the Flink SQL code for the tmall_d11 job can be executed directly. Otherwise, change the names of the DataHub topic and RDS table based on the topic and table that you have created. The sample code is as follows:

```
replace into tmall_trade_state
  select
    from_unixtime(FLOOR(tmall_trade_detail.gmtdate/1000), 'yyyy-MM-dd') as gmt_date,
    count(tid) as trade_count,
    sum(payment) as trade_sum
  from
    tmall_trade_detail
  group by
    from_unixtime(FLOOR(tmall_trade_detail.gmtdate/1000), 'yyyy-MM-dd');
```

Note You can modify the information about tables and fields as required.

3. Debug the Flink SQL code.

   The data during the Double 11 Shopping Festival is available for testing. To debug the code, download the test data and upload the data on the **Development** page for debugging.

4. Publish the SQL file for the tmall_d11 job.

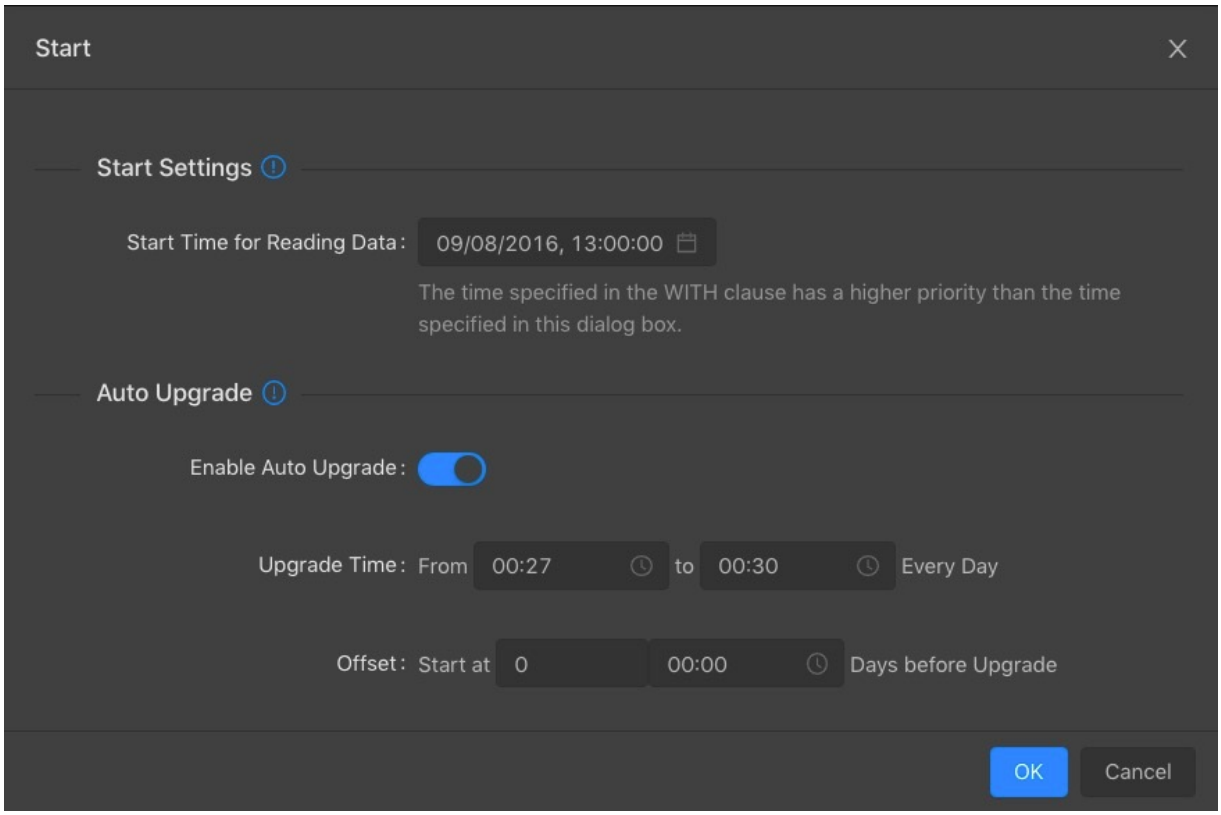
   After the computational logic has been verified in the debugging phase, click **Publish** on the **Development** page to publish the SQL file for the tmall_d11 job. Then, you can view the tmall_d11 job on the **Administration** page of the Realtime Compute console, and manage the job in the production environment, such as starting the job.

3.2.4.6. Administration

On the **Administration** page, you can click **Start** in the **Actions** column and specify the parameters on the page that appears to start a stream processing job, for example, the tmall_d11 job.

Note After you click **Start**, a dialog box is displayed. In the dialog box, you can specify the start time for reading data from the source data store.

The specified start time must be earlier than the file upload time. For example, the start time can be one hour earlier than the file upload time. In the Double 11 use case, the current time is 14:10, and 10 minutes have elapsed since the source data is uploaded. Therefore, the start time is set to 13:00.



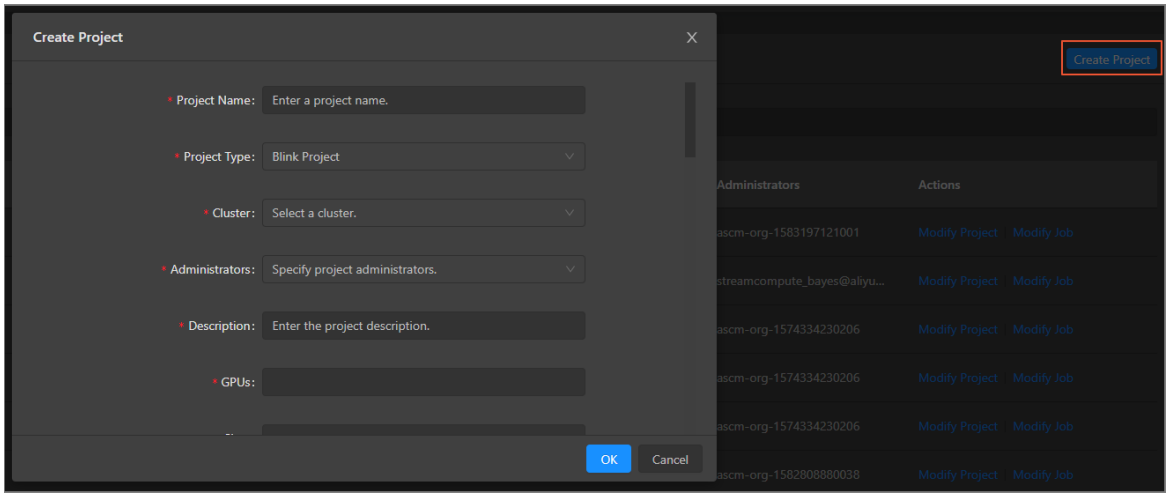
You can check the result data in the ApsaraDB for RDS data store after the job runs as expected. In the result table, five transactions and a turnover of CNY 500 are displayed. This is consistent with the source data for testing. In this way, an end-to-end verification is performed to check the SQL code.

3.3. Project management

This topic describes how to create and search for a project.

Create a project

1. Log on to the Realtime Compute console.
2. In the top navigation bar, move the pointer over your profile picture and click **Project Management**.
3. In the upper-right corner of the **Projects** page, click **Create Project**.
4. Configure the project information.



Parameter description

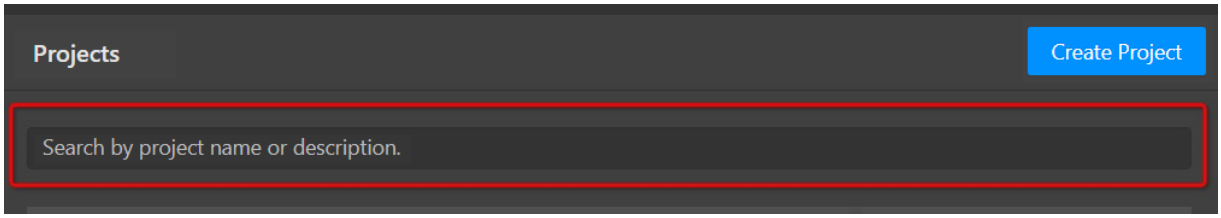
Parameter	Description
Project Name	The name of the project.
Project Type	The type of the project. **Blink Project** is selected by default.
Cluster	The cluster on which the jobs in the project run.
Description	The description of the project.
GPUs	The number of GPUs that are used by the project.
Slots	The number of compute units (CUs) that are used by the project. One CU is assigned with one CPU core and 4 GB memory.
Alert Methods	The methods in which alerts are sent when errors occur during job running. You can receive alerts by using text messages or TradeManager messages.
File Types	The supported file types. You can keep the default setting.
Storage Types	The supported data store types. You can keep the default setting.
Max Data Stores	The maximum number of data stores that can be registered in Realtime Compute. You can keep the default setting.
Max File Versions	The maximum number of code versions for an SQL file. You can keep the default setting.
Max Folders	The maximum number of folders that can be created in the project. You can keep the default setting.
Max Folder Levels	The maximum number of folder levels in the project. You can keep the default setting.
Max Files	The maximum number of job SQL files that can be created in the project. You can keep the default setting.

Parameter	Description
Max Resources	The maximum number of JAR files and DICTIONARY resources that can be uploaded. You can keep the default setting.
Max Referenced Resources	The maximum number of JAR files and DICTIONARY resources that can be referenced. You can keep the default setting.
Monitoring and Alerting	Specifies whether to enable the monitoring and alerting feature. You can keep the default setting.
Data Collection	Specifies whether to collect data while a job is running. You can keep the default setting.
Data Display	Specifies whether to display data. You can keep the default setting.
Metastore	Specifies whether to display metadata. You can keep the default setting.
Data Storage	Specifies whether to enable data store registration. This feature is enabled by default. You can keep the default setting.
Engine	Specifies whether to display the engine. You can keep the default setting.
Online Logs	Specifies whether to record the job running logs. This feature is enabled by default. You can keep the default setting.
Resource Management	Specifies whether resources such as JAR files can be uploaded. This feature is enabled by default. You can keep the default setting.
Switch Version	Specifies whether to enable the job version switch feature. This feature is enabled by default. You can keep the default setting.
Project Protection	Specifies whether to enable the project lock feature. You can keep the default setting.

5. Click **OK**.

Search for a project

On the **Projects** page, enter a keyword or full name of a project in the search box to find the project.



3.4. Data storage

3.4.1. Overview

This chapter describes data storage systems supported by Realtime Compute.

3.4.2. Overview

3.4.2.1. Overview

To facilitate data storage management, you can register data storage resources on the Realtime Compute development platform. This enables you to leverage the advantages of the one-stop Realtime Compute service. In Realtime Compute, you can manage multiple data storage systems, such ApsaraDB for RDS, and Table Store. With this one-stop management service, you no longer have to navigate across multiple management consoles of different storage systems.

3.4.2.2. Types

Realtime Compute supports both streaming data storage and static data storage.

Streaming data storage

Streaming data storage systems provide inputs and outputs for downstream Realtime Compute jobs.

Streaming data storage

Storage system	Input	Output
DataHub	Supported	Supported
Log Service	Supported	Supported
MQ	Supported	Supported

Static data storage

Static data storage systems provide outputs for downstream Realtime Compute jobs and allow you to perform association queries.

Static data storage

Storage system	Dimension table	Output
ApsaraDB for RDS	Supported	Supported
Table Store	Supported	Supported

3.4.2.3. Registration and usage

This topic describes how to register and use external data stores in Realtime Compute.

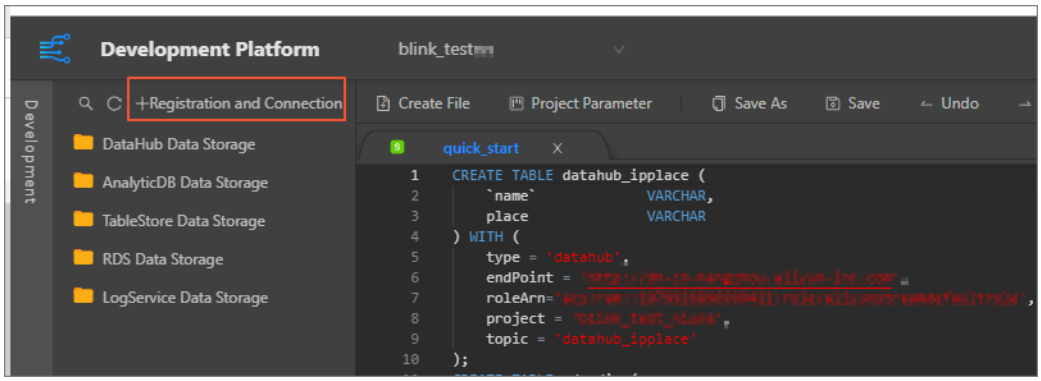
> **Note** If a job requires the use of the data stores owned by another Apsara Stack tenant account, you can write DDL statements to reference the data stores. In the DDL statements, you must specify the AccessKey ID and AccessKey secret of the account. In this scenario, you cannot use the codeless UI to manage the data stores.

Register a data store

To register a data store, follow these steps:

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane of the page that appears, click the **Storage** tab. Select the folder for the data store that you want to register. Then, click **+Registration and Connection**.

   Register a data store



4. In the dialog box that appears, configure the required parameters and click **OK**.

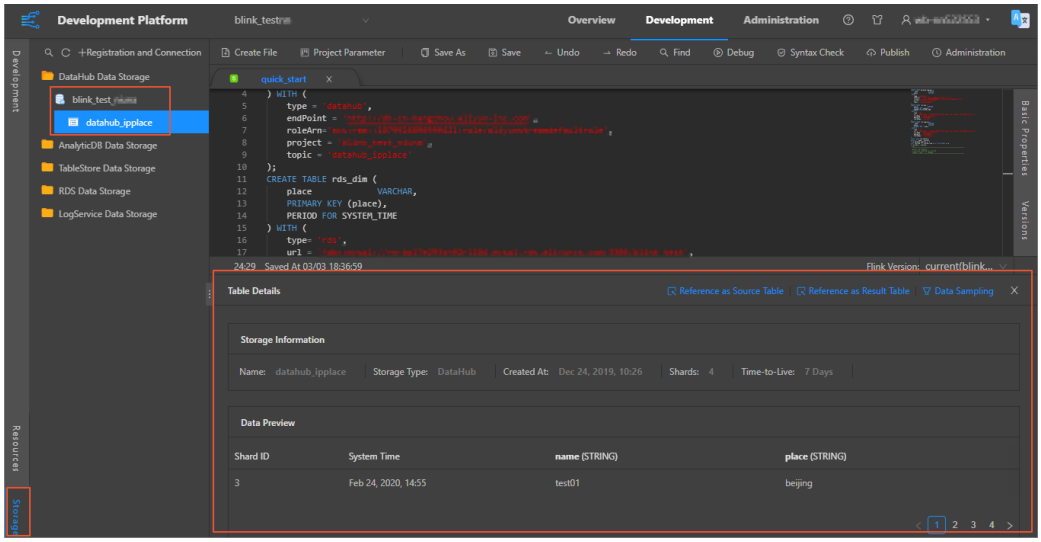
> **Note** After you enable the data store registration feature, you can only register data stores that are owned by your organization.

Preview data from a data store

Realtime Compute provides the data preview feature for each registered data store. To preview data, click the **Storage** tab and double-click the folder of the target data store in the left-side navigation pane. The following example shows how to preview data from a DataHub data store.

1. Log on to the Realtime Compute console. In the top-navigation bar, click Development. On the page that appears, click the **Storage** tab and double-click the **DataHub Data Storage** folder.
2. Double-click the target project and then the target topic to view the details.

   Table details



Use the auto DDL generation feature

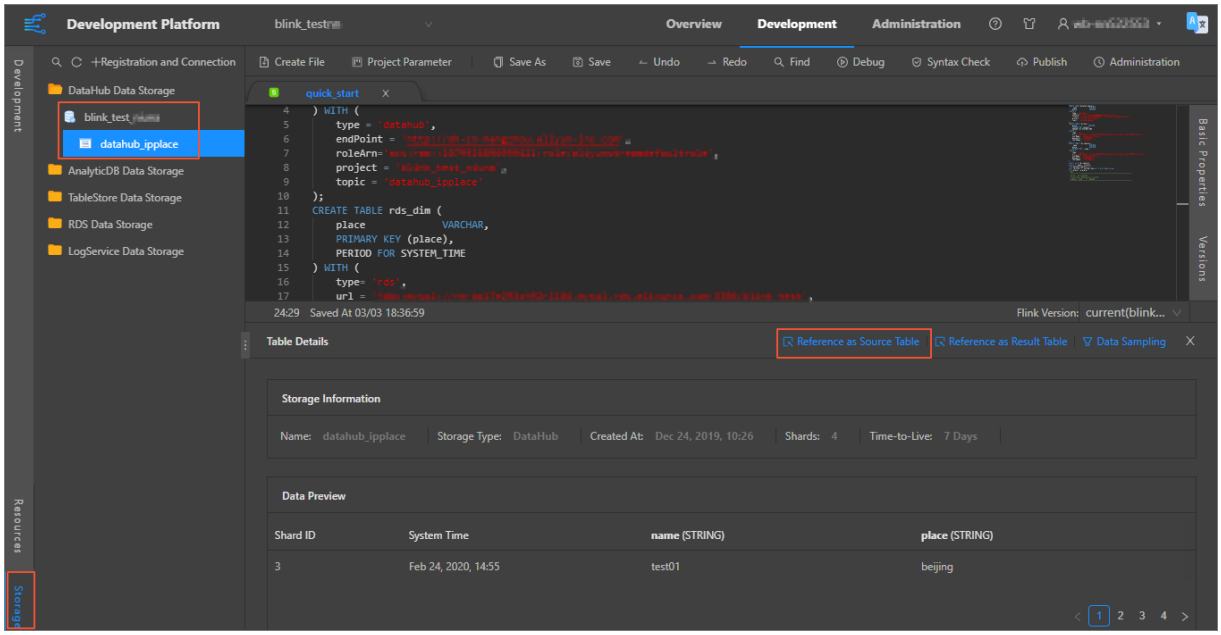
You must declare tables from external data stores before you can reference these tables. The following example shows how to reference a source table that contains streaming data:

```
CREATE TABLE in_stream( a varchar, b varchar, c timeStamp) with ( type='datahub', endPoint='http://dh-cn-hangzhou.aliyuncs.com', project='blink_test', topic='ip_count02', accessId='LTAIYtaf******', accessKey='gUqyVwfkK2vfJI7jF90******');
```

The field names in the table that is referenced on the Development page must be the same as those in the DataHub topic. You must declare the field data types in the code based on the field type mapping between DataHub and Realtime Compute to ensure that Realtime Compute can identify the data. Realtime Compute offers the auto DDL generation feature. The following section describes how to use this feature.

1. In the left-side navigation pane of the **Development** page, click the **Storage** tab.
2. On the **Storage** tab, navigate through cascaded folders and nodes to find the target table. Then, double-click the name of the target table.
3. In the **Table Details** pane that appears, click **Reference as Source Table**, **Reference as Result Table**, or **Reference as Dimension Table** as required. Then, you can obtain the DDL statements that are automatically generated to reference the target table.

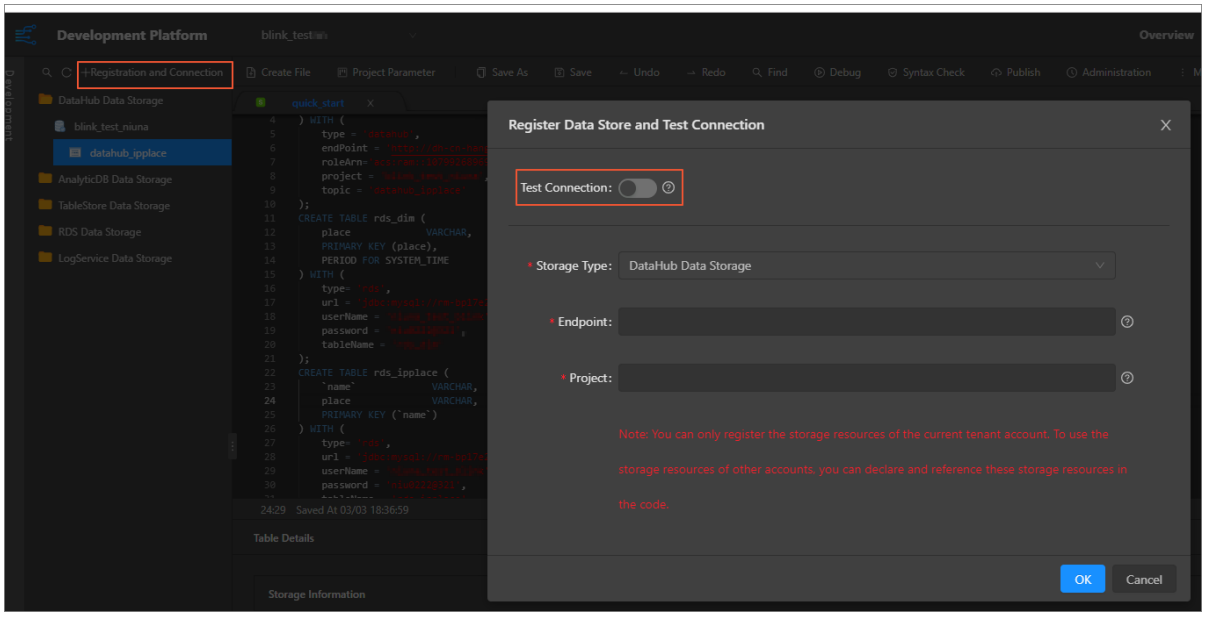
To reference a source table, log on to the Realtime Compute console and open the target SQL file on the Development page. Click the Storage tab, select the table for reference, and then click **Reference as Source Table**. The required DDL statements are displayed on the current page.



Test network connection

Realtime Compute offers the network connection test feature for data stores. This feature allows you to test the connection between Realtime Compute and a target data store. To enable the network connection test feature, follow these steps:

1. In the left-side navigation pane of the **Development** page, click the **Storage** tab.
2. In the upper-right corner of the **Storage** tab, click **+Registration and Connection**.
3. In the **Register Data Store and Test Connection** dialog box, turn on **Test Connection**.



Example: Reference data stores owned by another level-1 organization

You can only register and use data stores that are owned by your level-1 organization. To use data stores that are owned by another level-1 organization, write DDL statements to create a reference to these data stores. For example, if a user from Organization A wants to use the data stores owned by Organization B, the user can enter the following DDL statements:

```
CREATE TABLE in_stream( a varchar, b varchar, c timeStamp) with ( type='datahub', endPoint='http://dh-cn-hangzhou.aliyuncs.com', project='blink_test', topic='ip_count02', accessId='AccessKey ID authorized by Organization B users', accessKey='AccessKey secret authorized by Organization B users');
```

3.4.3. Register a DataHub data store

DataHub, an Alibaba Cloud streaming data service, is a real-time data distribution platform designed to process streaming data. You can publish and subscribe to applications for streaming data in DataHub and distribute the data to other platforms. DataHub allows you to analyze streaming data and build applications based on the streaming data. Realtime Compute often uses DataHub to store source and result tables that contain streaming data.

Register a DataHub project

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click the **Storage** tab.
4. Right-click the **DataHub Data Storage** folder and select **Register Data Store** to register a DataHub project in Realtime Compute.

Parameter description

Parameter	Description
Test Connection	Specifies whether to enable the network connection test feature. Network connection tests are automatically performed on data stores that can be registered in Realtime Compute. To test the connection between Realtime Compute and data stores that cannot be registered, turn on the Test Connection switch.
Storage Type	The type of the data store. DataHub Data Storage is selected by default.
Endpoint	The endpoint of DataHub. The endpoint of DataHub varies with regions. For more information about endpoints, contact your administrator. **Note** To specify this parameter for Apsara Stack, contact your Apsara Stack administrator to obtain the endpoint of DataHub.
Project	The name of the DataHub project. **Note** You can only register DataHub projects that are owned by your level-1 organization. For example, if DataHub Project A is owned by Organization A, users from Organization B cannot register Project A in Realtime Compute.

Parameter	Description
AccessKey ID	The AccessKey ID of the current account.
AccessKey Secret	The AccessKey secret of the current account. The AccessKey secret enables Realtime Compute to access the DataHub project.

Scenarios

DataHub is a streaming data storage system that can be used to store source and result tables. However, it cannot be used to store dimension tables for Realtime Compute.

FAQ

Q: Why am I unable to register a DataHub project in Realtime Compute?

A: Realtime Compute uses a storage software development kit (SDK) to access different data stores. The Storage tab in the Realtime Compute console only helps you manage data from different data stores. You can perform the following operations to troubleshoot registration errors:

- Check whether you have created a DataHub project and have the permissions to access the project. You can log on to the DataHub console and check whether you can access the project.
- Check whether you are the project owner. You can only register DataHub projects that are owned by your level-1 organization. For example, if DataHub Project A is owned by Organization A, users from Organization B cannot register Project A in Realtime Compute.
- Check whether you have specified the correct DataHub endpoint and project name.
- Check whether you have specified a classic network endpoint for the Endpoint parameter. If you specify a VPC endpoint, the DataHub project will fail to be registered.
- Check whether you have registered the DataHub project. Realtime Compute provides a registration check mechanism to prevent duplicate registration.

Q: Why does Realtime Compute only support time-based sampling?

A: DataHub stores streaming data, and you can only specify time parameters in the API. Therefore, Realtime Compute supports only time-based sampling.

3.4.4. Register a Log Service data store

Log Service (previously known as SLS) provides an end-to-end solution for log management. You can use Log Service to easily collect, subscribe to, dump, and query large amounts of log data. Realtime Compute can integrate with Log Service to process logs. This eliminates the need of data migration.

Register a Log Service project

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click the **Storage** tab.
4. Right-click the **LogService Data Storage** folder and select **Register Data Store** to register a Log Service project in Realtime Compute.

Parameter description

Parameter	Description
Test Connection	Specifies whether to enable the network connection test feature. Network connection tests are automatically performed on data stores that can be registered in Realtime Compute. To test the connection between Realtime Compute and data stores that cannot be registered, turn on the Test Connection switch.
Storage Type	The type of the data store. LogService Data Storage is selected by default.
Endpoint	The endpoint of Log Service. The endpoint of Log Service varies with regions. Note For more information about the endpoint of Log Service, contact the Apsara Stack system administrator.
Project	The name of the Log Service project. Note You can only register Log Service projects that are owned by your level-1 organization. For example, if Log Service Project A is owned by Organization A, users from Organization B cannot register Project A in Realtime Compute.
AccessKey ID	The AccessKey ID of the current account.
AccessKey Secret	The AccessKey secret of the current account. The AccessKey secret enables Realtime Compute to access the Log Service project.

Scenarios

Log Service is a streaming data storage system that can be used to store source tables and result tables. However, it cannot be used to store dimension tables for Realtime Compute.

FAQ

- Q: Why am I unable to register a Log Service project in Realtime Compute?

  A: Realtime Compute uses a storage software development kit (SDK) to access different data stores. The Storage tab in the Realtime Compute console only helps you manage data from different data stores. You can perform the following operations to troubleshoot registration errors:

  - Check whether you have created a Log Service project and have the permissions to access the project. You can log on to the Log Service console and check whether you can access the project.

- Check whether you are the project owner. You can only register Log Service projects that are owned by your level-1 organization. For example, if Log Service Project A is owned by Organization A, a user from Organization B cannot register Project A in Realtime Compute.
- Check whether you have specified the correct Log Service endpoint and project name.

  > **Note** The endpoint must start with http and cannot end with a forward slash (/). For example, *http://cn-hangzhou.log.aliyuncs.com* is correct, and *http://cn-hangzhou.log.aliyuncs.com/* is incorrect.

- Check whether you have registered the Log Service project. Realtime Compute provides a registration check mechanism that prevents duplicate registration.

- Q: Why does Realtime Compute support only time-based sampling?

  A: Log Service stores streaming data, and you can only specify time parameters in the API. Therefore, Realtime Compute supports only time-based sampling.

  > **Note** To use the search feature of Log Service, log on to the Log Service console.

3.4.5. Register a Tablestore data store

Tablestore is a NoSQL database service that is based on the Apsara distributed system. Tablestore allows you to store and access large amounts of structured data in real time. Tablestore features massive data storage and low access delays, which makes it suitable to store dimension tables and result tables for Realtime Compute.

Register a Tablestore instance

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click the **Storage** tab.
4. Right-click **TableStore Data Storage** and select **Register Data Store**. In the dialog box that appears, register a Tablestore instance in Realtime Compute.

Parameters

Parameter	Description
Test Connection	Specifies whether to enable the network connection test feature. Network connection tests are automatically performed on data stores that can be registered in Realtime Compute. To test the connection between Realtime Compute and data stores that cannot be registered, turn on the **Test Connection** switch.
Storage Type	The type of the data store. **TableStore Data Storage** is selected by default.
Endpoint	The endpoint of the Tablestore instance. You must enter the internal endpoint of the Tablestore instance. You can log on to the Tablestore console to view the internal endpoint of the instance.

Parameter	Description
Instance Name	The name of the Tablestore instance.
AccessKey ID	The AccessKey ID of the current account.
AccessKey Secret	The AccessKey secret of the current account. The AccessKey secret enables Realtime Compute to access the Tablestore instance.

3.4.6. Register an RDS data store

This topic describes how to register and use an ApsaraDB for RDS data store in Realtime Compute.

RDS overview

RDS offers a stable, reliable, and scalable online database service. Based on the Apsara distributed system and high performance SSD storage, RDS supports a wide range of engines, such as MySQL, PostgreSQL, and Postgres Plus Advanced Server (PPAS, highly compatible with Oracle). Realtime Compute supports the following RDS engines: MySQL and PostgreSQL.

The performance of Tablestore in high concurrency scenarios where large amounts of data need to be processed is higher than that of RDS. The performance of RDS is restricted by the limits of relational models. Therefore, RDS is often used to store result tables for Realtime Compute. In low concurrency scenarios where a small number of data needs to be processed, RDS can be used to store dimension tables.

> **Note** Realtime Compute uses relational databases, such as ApsaraDB RDS for MySQL, to store result data. Distributed Relational Database Service (DRDS) and RDS connectors are used. If Realtime Compute frequently writes data into a DRDS or RDS table, deadlocks may occur. In scenarios that require high queries per second (QPS), high transactions per second (TPS), or highly concurrent write operations, we recommend that you do not use DRDS or RDS to store the result tables of Blink jobs. To prevent deadlocks, we recommend that you use Tablestore to store result tables.

Register an RDS instance

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click the **Storage** tab.
4. Right-click **RDS Data Storage**, and select **Register Data Store**. In the dialog box that appears, register an RDS instance in Realtime Compute.

Parameters

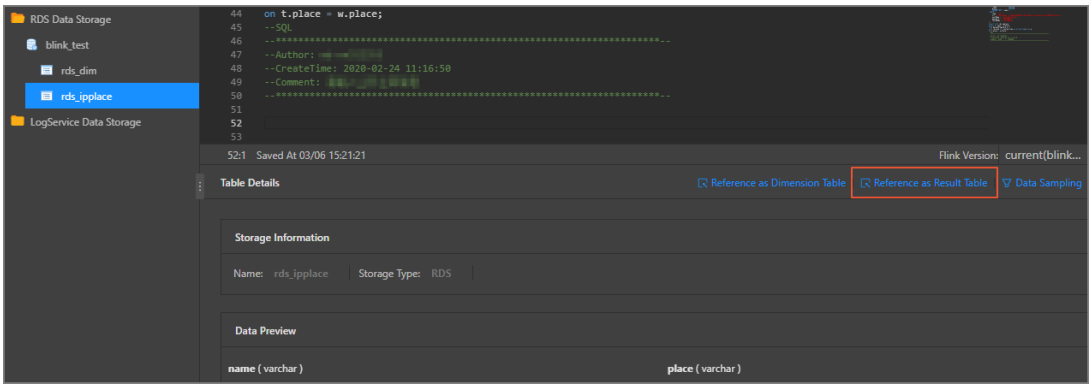
Parameter	Description

Parameter	Description
Test Connection	Specifies whether to enable the network connection test feature. Network connection tests are automatically performed on data stores that can be registered in Realtime Compute. To test the connection between Realtime Compute and data stores that cannot be registered, turn on the **Test Connection** switch.
Storage Type	The type of the data store. **RDS Data Storage** is selected by default.
URL	The connection URL of the RDS database.
DBName	The name of the RDS database to be accessed by Realtime Compute. **Note** This parameter specifies the RDS database name instead of the RDS instance name. RDS uses whitelists for access control to ensure system security. The IP addresses of the Realtime Compute console and worker nodes must be added to RDS whitelists. Otherwise, Realtime Compute may fail to connect to RDS. For more information, see Specify whitelist settings.
User Name	The username that you use to log on to the RDS database.
Password	The password that you use to log on to the RDS database.
Engine	The type of the RDS database. Valid values: ○ **mysql** ○ **postgresql** ○ **sqlserver**

Reference an RDS table as a result table

After you register an RDS data store, double-click the RDS database, double-click the RDS table that you want to reference as a result table, and then click **Reference as Result Table**.

Reference an RDS table as a result table



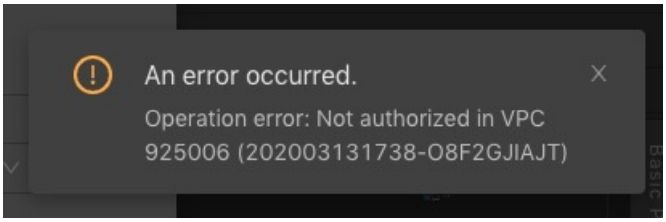
After you click **Reference as Result Table**, Realtime Compute automatically generates the corresponding DDL statements on the current page.

Result

```
31      place             VARCHAR,
32      PRIMARY KEY (`name`)
33  ) WITH (
34      type= 'rds',
35      url = '[redacted]',
36      userName = '[redacted]',
37      password = '[redacted]',
38      tableName = '[redacted]'
39  );
```

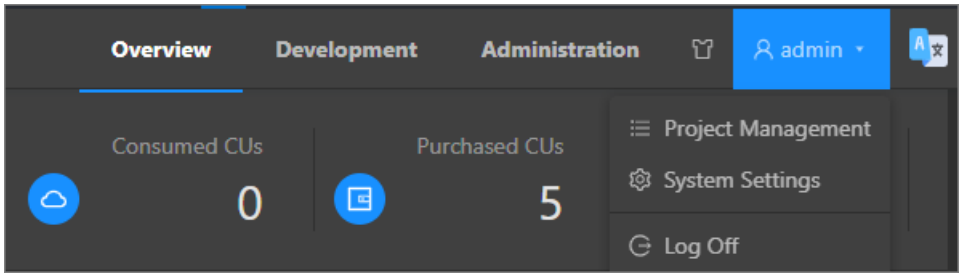
If the following error message appears, troubleshoot and rectify the fault as follows.

Error message



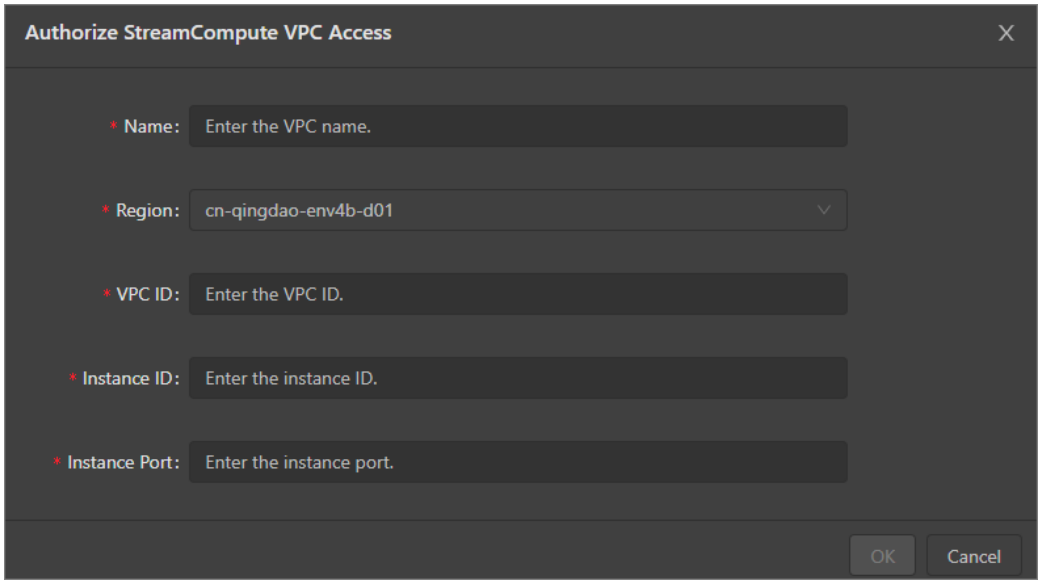
The error occurs because a VPC (not a classic network) was selected when you created the RDS instance. To rectify this fault, follow these steps:

1. Move the pointer over the administrator icon, as shown in the following figure.

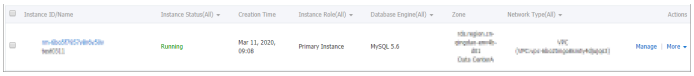


2. Click **System Settings**.
3. In the left-side navigation pane, click **VPC Access Authorization**.
4. Click **Add Authorization**. The **Authorize StreamCompute VPC Access** page appears.

Authorization



Parameters

Parameter	Description
Name	The name of the VPC.
Region	The region where RDS resides.
VPC ID	The ID of the VPC.
Instance ID	The ID of the RDS instance. You can log on to the RDS console and view the instance ID. Instance information 
Instance Port	The access port for the RDS instance. To view the internal port number, log on to the RDS console, click the target instance ID or name in the **Instance ID/Name** column. On the page that appears, view the internal port number in the Basic Information section.

5. Register the target RDS instance. You must specify the required parameters during the registration.

You can only register data storage resources that are owned by your level-1 organization. For example, if RDS Instance A is owned by Organization A, a user from Organization B cannot register RDS Instance A in the Realtime Compute console. To use Instance A in a stream processing job, the user from Organization B must use SQL code to create a reference to Instance A.

> **Note** If you want to use the RDS storage resources owned by your level-1 organization, we recommend that you do not use SQL code to create a reference to these resources.

The user from Organization B must also specify the following parameters in the WITH clause based on the information of Instance A: url, userName, password, and tableName.

Configuration

```
CREATE TABLE datahub_output (
    id            varchar
) WITH (
    type= 'rds',
    url = '[redacted]',
    userName = '[redacted]',
    password = '[redacted]',
    tableName = 'datahub_output'
);
```

To use RDS storage resources by writing SQL code, the user from Organization B must specify whitelist settings.

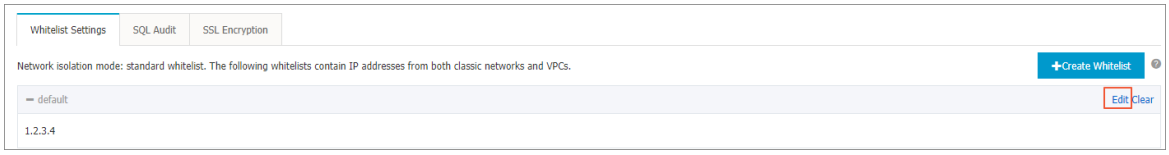
Specify whitelist settings

Some data stores use whitelists for access control to ensure high-level security. These data stores allow access only from the IP addresses that are added to the whitelists. This prevents unauthorized Apsara Stack services from accessing data in these data stores. For example, a newly created RDS database denies all access. You must add IP addresses to the RDS database whitelists to allow access to the database.

RDS can be accessed from both external and internal networks. To enable Realtime Compute to access RDS, you must add the CIDR blocks of Realtime Compute to RDS whitelists.

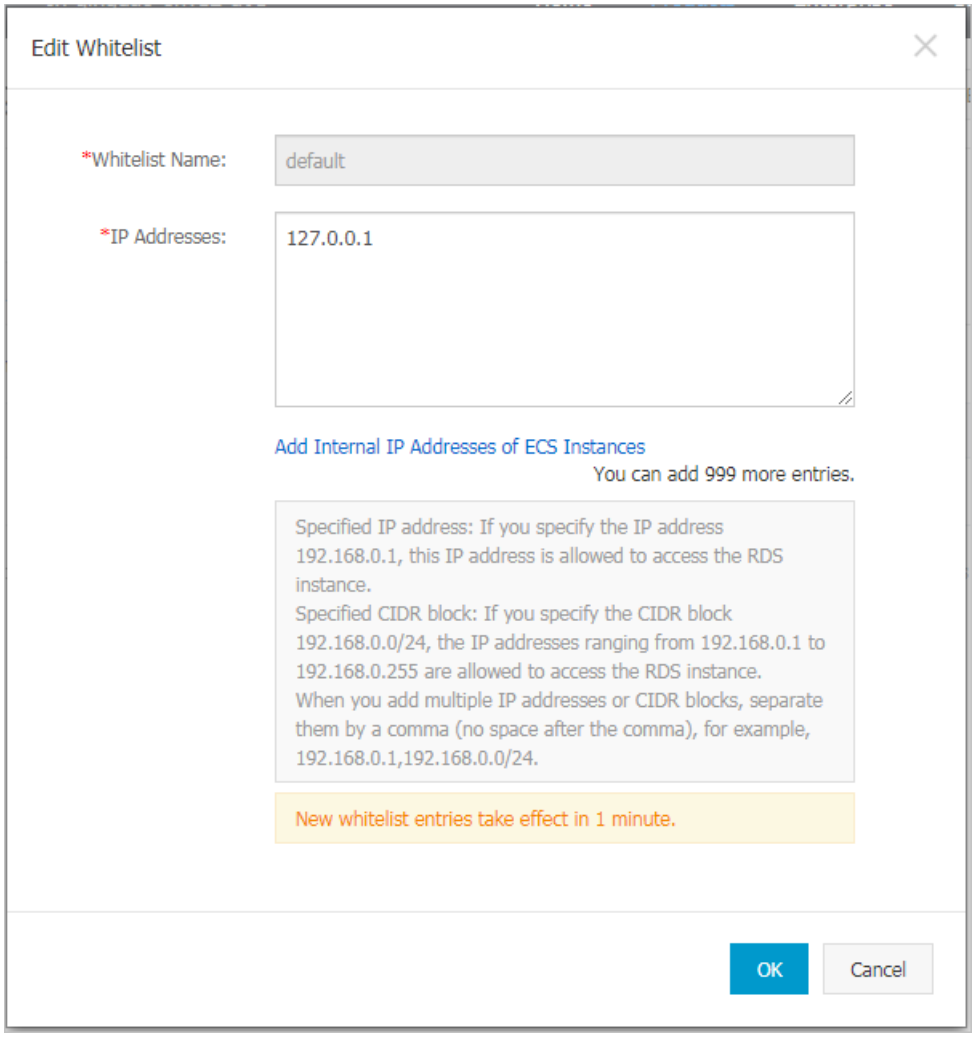
To do this, follow these steps:

1. Log on to the ApsaraDB for RDS console.
2. On the **Instances** page, click the ID of an instance in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Data Security**.
4. On the **Whitelist Settings** tab, click **Edit** corresponding to the **default** whitelist.



> **Note**
>
> - To connect an ECS instance to an RDS instance by using an internal endpoint, you must make sure that the two instances are in the same region and have the same network type. Otherwise, the connection fails.
> - You can also click **Create Whitelist** to create a new whitelist.

5. In the **Edit Whitelist** dialog box that appears, specify the IP addresses or CIDR blocks that are used to access the instance, and then click **OK**.



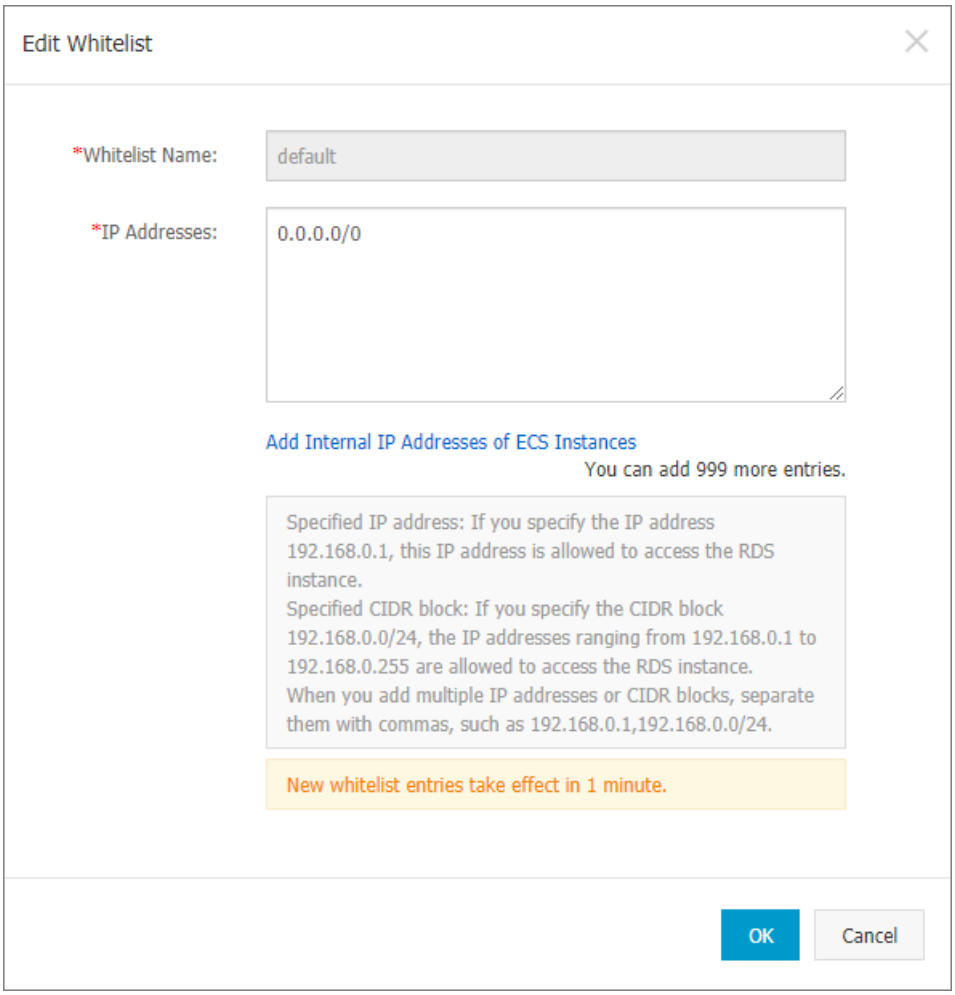
- If you specify the 10.10.10.0/24 CIDR block, IP addresses in the 10.10.10.X format are allowed to access the RDS instance.
- If you want to add multiple IP addresses or CIDR blocks, separate entries with commas (without spaces), such as 192.168.0.1,172.16.213.9.
- After you click **Add Internal IP Addresses of ECS Instances**, the IP addresses of all the ECS instances under your Apsara Stack account are displayed. You can select the required IP addresses to add to the whitelist.

> **Note** If you add a new IP address or CIDR block to the **default** whitelist, the default address 127.0.0.1 is automatically deleted.

Troubleshooting

- Fault description

A stack exception occurs while the system is running, as shown in the following figure.



- Solution

  Add the IP address of your region to an RDS whitelist. For more information, see Specify whitelist settings.

3.5. Data development

3.5.1. Create a job

This topic describes how to create a Realtime Compute job.

Procedure

1. Log on to the Realtime Compute console.
2. Click **Development Platform**.
3. Click **Development** in the top navigation bar.
4. Click **Create File** in the toolbar.
5. In the **Create File** dialog box, specify the required fields.

Field	Description
File Name	The name of the file. The specified name must be 3 to 64 characters in length and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter.
File Type	The type of the file. Valid values: FLINK_STREAM/SQL and FLINK_STREAM/DATASTREAM.
Storage Path	The folder of the file. You can click the icon on the right side of an existing folder and create a subfolder.

6. Click **OK**.

3.5.2. Development

3.5.2.1. SQL code assistance

The development platform of Realtime Compute offers a complete set of SQL tools in the integrated development environment (IDE). These tools provide the following features to help you with Flink SQL-based development:

- Syntax check

  On the **Development** page of Realtime Compute, the revised script is automatically saved. When the script is saved, an SQL syntax check is automatically performed. If a syntax error is detected, the **Development** page shows the row and column where the error is located, and the cause of the error.

- Intelligent code completion

  When you enter SQL statements on the **Development** page of Realtime Compute, auto completion popups about keywords, built-in functions, tables, or fields are automatically displayed.

- Syntax highlighting

  Flink SQL keywords are highlighted in different colors to differentiate data structures.

3.5.2.2. SQL code version management

Realtime Compute provides key features that help you complete development tasks, such as coding assistance and code version management. A new code version is generated each time you publish a job SQL file. The code version management feature allows you to track code changes and roll back to an earlier version if required.

- Manage code versions

  On the **Development** page, you can manage Flink SQL code versions. A new code version is generated each time you publish a job SQL file. You can use the code version management feature to track versions, modify the code, and roll the code back to an earlier version.

  On the **Versions** tab on the right side of the **Development** page, click **More** in the **Actions** column to manage code versions.

  - **Compare**: Check the differences between the current version and an earlier version.

- **Rollback**: Roll back to an earlier version.
- **Delete**: Delete an earlier version.
- **Locked**: Lock the current version.

> **Note** You cannot submit a new version before you unlock the SQL file.

- Delete code versions

  A snapshot of a code version is created each time you submit an SQL file for publishing a job. This allows you to track code changes. The maximum number of code versions has been specified. If you use Apsara Stack, a maximum of 20 code versions can be published. To find out the maximum number of code versions in other environments, contact the system administrator. If the number of code versions reaches the upper limit, an error message is displayed to alert you to delete one or more earlier versions.

  In this scenario, you must delete one or more earlier versions before you publish new versions. To do this, click the **Versions** tab on the right side of the **Development** page, click **More** in the **Actions** column, and select **Delete** to delete expired versions that are no longer needed.

3.5.2.3. Data store management

The Development page of the Realtime Compute console provides an easy and effective method to manage data stores. For example, you can register external data stores to reference the data stores.

- Data preview

  The **Development** page of Realtime Compute allows you to preview data from a wide range of data stores. Data preview allows you to analyze the characteristics of upstream and downstream data, identify key business logic, and complete development tasks with high efficiency.

- Auto DDL generation

  Realtime Compute provides the auto DDL generation feature. The system can automatically generate DDL statements to reference data stores that can be registered in Realtime Compute. This feature provides a simple method to edit SQL statements for stream processing jobs. This improves overall efficiency and reduces errors when you manually enter SQL statements.

3.5.3. Debug job code

The Realtime Compute development platform provides a simulated running environment where you can customize uploaded data, simulate operations, and check outputs.

After you write the SQL code that implements the computing logic, follow these steps to debug the code:

1. Log on to the Realtime Compute console.
2. In the top navigation bar, click **Development**.
3. In the left-side navigation pane, click **Development**.
4. On the **Development** tab, double-click the target folder and file name to open the job file.
5. In the top menu bar, click **Syntax Check**.

> **Note** You can use the syntax check feature to check whether the SQL file includes syntax errors. Error messages are displayed for syntax errors.

6. In the top menu bar, click **Debug**. On the **Debug File** page, debug your SQL code.

   The test data for debugging can be acquired by using either of the following two methods:

   - Upload local data.
     a. Click **Download Template**.
     b. Prepare test data based on the template.
     c. Click **Upload**. After the file is uploaded, you can view the uploaded data in the data preview section.
   - Sample online data.
     a. Click **Random Online Data Sampling** or **Sequential Online Data Sampling**.
     b. View the sampled data in the data preview section.

7. Click **OK**.
8. In the output window, view the debugging result.

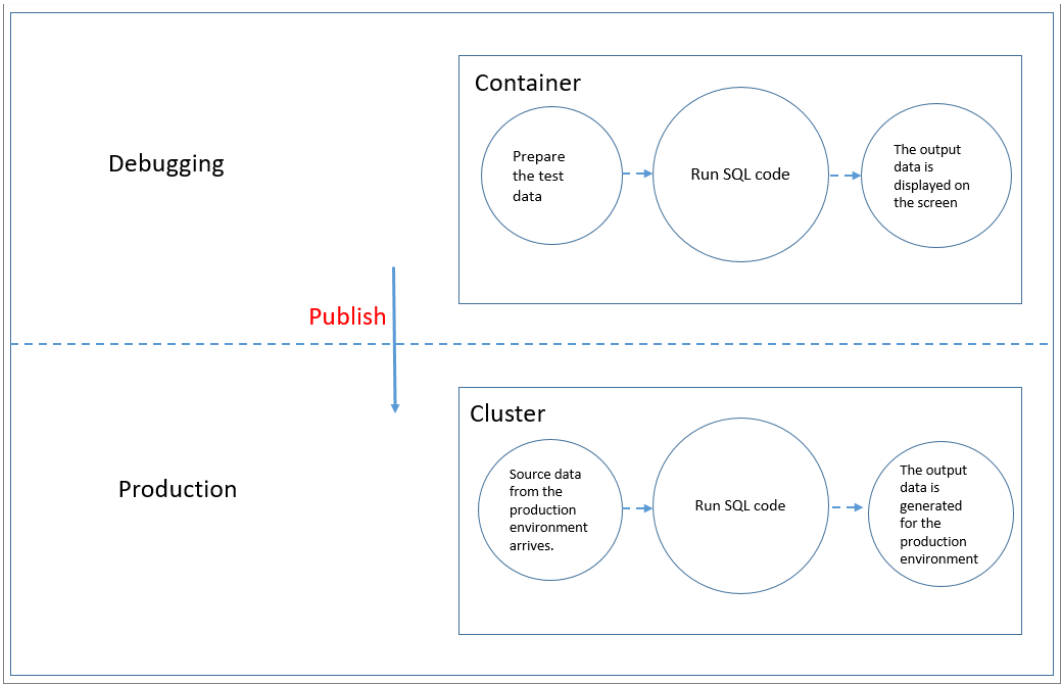
The debugging feature of Realtime Compute provides the following functions:

- Enables isolation between debugging and production environments.

  In the debugging environment, the Flink SQL code runs in a separate container, and computing result data is only displayed on the screen of the **Development** page. In this way, the debugging does not affect the running jobs and data stores in the production environment.

  In the debugging phase, result data is not written to external data stores. In the production environment, failures may occur due to format errors when result data is written to the target data stores. Such failures cannot be identified or prevented in the debugging phase, and can be detected only while jobs are running. For example, failures may occur in the production environment if your result data is too long. This occurs when the result data is written to a result table in ApsaraDB for RDS and the length of character strings reaches the upper limit for an RDS table. The Realtime Compute team is working on support for writing result data to external data stores in the production environment. This allows you to effectively simulate the production environment and resolve more issues in the debugging phase.

  Isolation between debugging and production environments



- Supports the customization of test data.

  In the debugging environment, Realtime Compute does not read data from source data stores, such as DataHub topics that store source tables and RDS instances that store dimension tables. You must create a set of test data and upload the test data on the **Development** page.

  To make the debugging feature easy to use, Realtime Compute provides a template of test data for each type of job. You can download the template and enter your test data.

  > **Note** We recommend that you use the templates to prevent errors.

- Specifies a separator.

  A comma (,) is used as the separator in files for debugging by default. An example of a file for debugging is described as follows:

  ```
  id,name,age
  1,alicloud,13
  2,stream,1
  ```

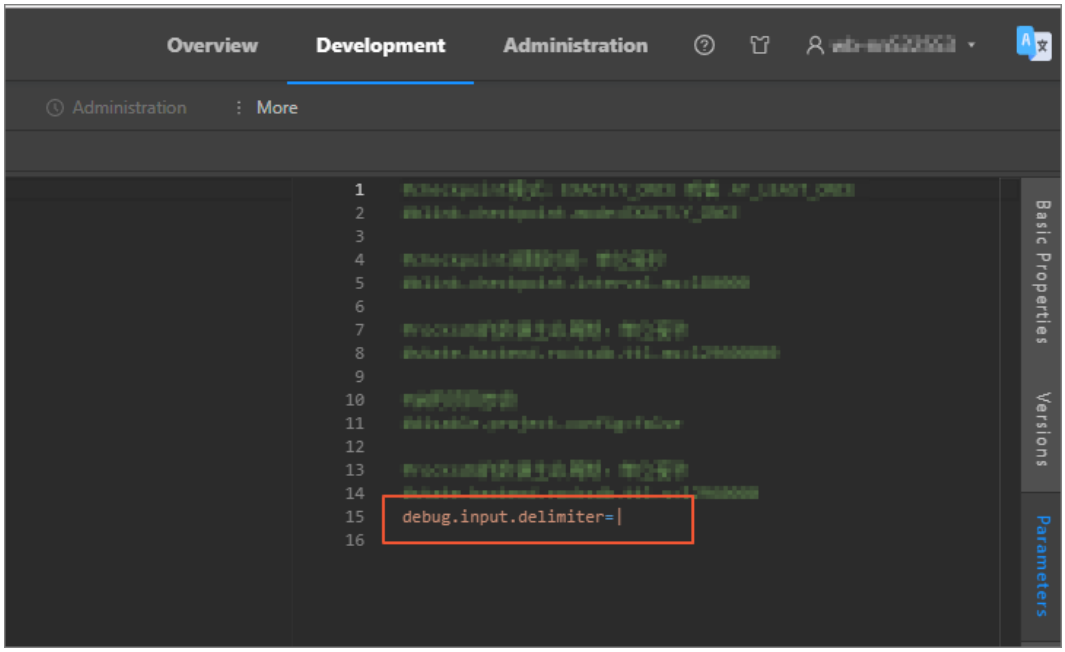
  If the separator is not specified, a comma (,) is used to separate fields. If you need to use a JSON string as the field data and the string contains commas (,), you must specify another character as the separator.

  > **Note** Realtime Compute allows you to specify a character as the separator, but not a multi-character string, such as aaa.

  ```
  id|name|age
  1|alicloud|13
  2|stream|1
  ```

In this example, specify the debug.input.delimiter parameter as follows: `debug.input.delimiter=|`.

Specify a separator



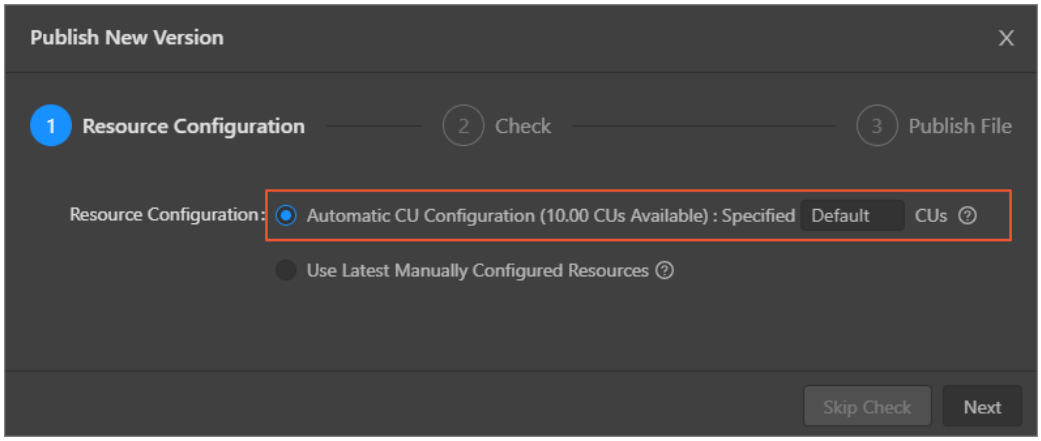
3.5.4. Publish a job SQL file

After you have created and debugged a job Flink SQL file, you can publish the SQL file and manage the job in the production environment.

Procedure

1. Log on to the Realtime Compute console.
2. In the top navigation bar, Click **Development**.
3. In the top menu bar, click **Publish**.
4. In the dialog box that appears, select **Automatic CU Configuration**. If you are performing automatic configuration for the first time, we recommend that you use the default number of CUs. Click **Next**.

   Configure resources



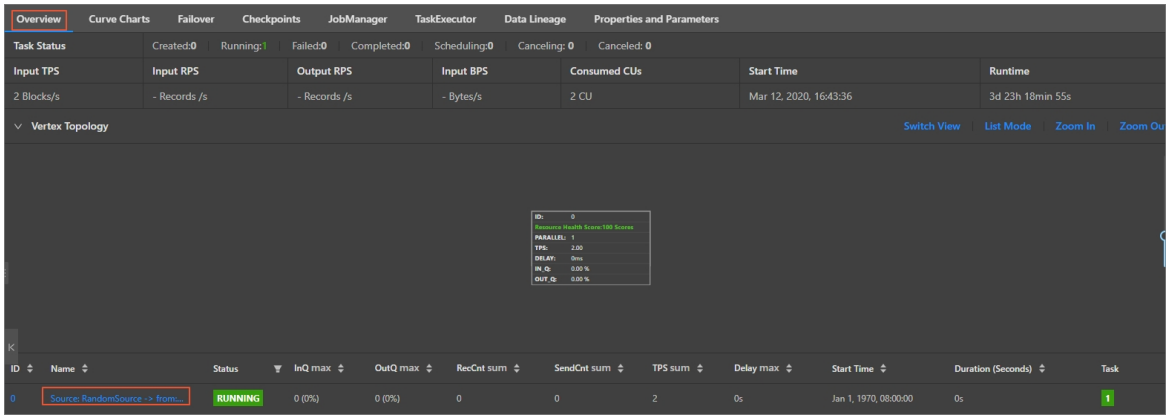
5. Check the data. After the check is completed, click **Next**.
6. Click **Publish**.
7. Go to the **Administration** page to start the job.
   i. In the top navigation bar, click **Administration**.
   ii. On the **Administration** page, find the target job, and click **Start** in the **Actions** column.

3.5.5. View logs

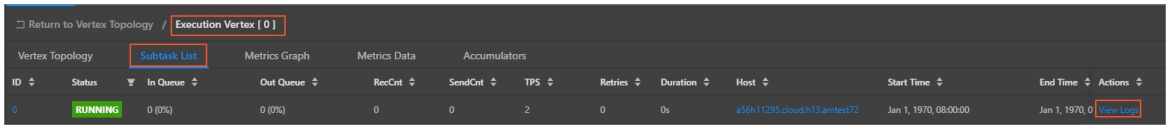
You can view job logs to check the job run information. This topic describes how to view job logs.

Procedure

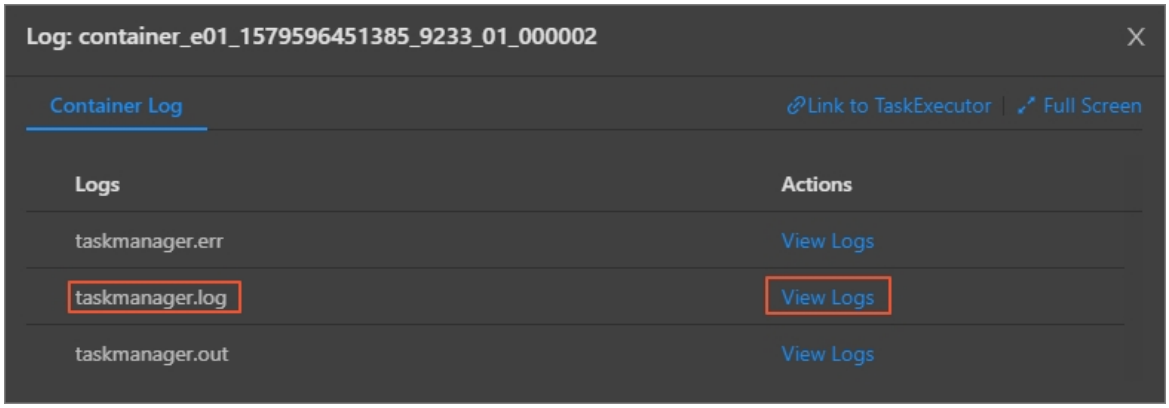
1. Go to the **Administration** page in Realtime Compute.
   i. Log on to the Realtime Compute console.
   ii. In the top navigation bar, click **Administration**.
   iii. On the **Jobs** page that appears, click the target job name in the **Job Name** column.
2. At the bottom of the **Overview** page, click the name of the target vertex.



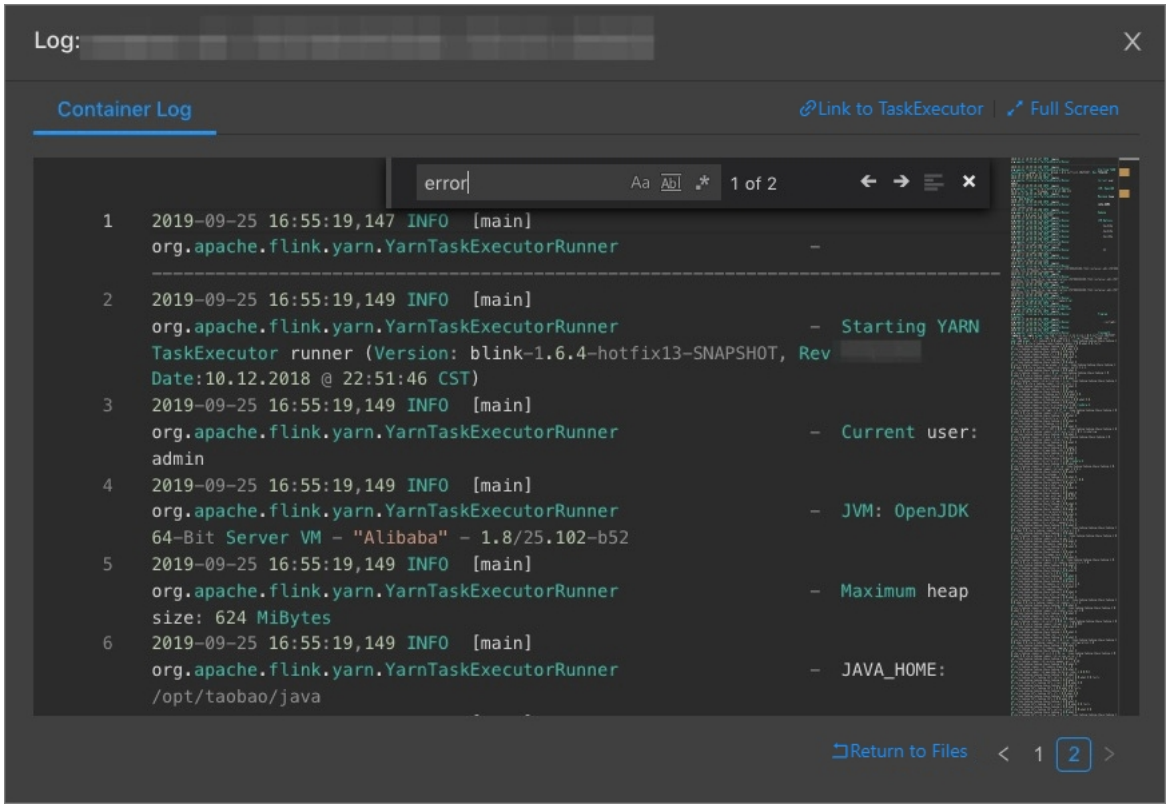
3. On the **Execution Vertex** page, click the **Subtask List** tab, and click **View Logs** in the **Actions** column.



4. In the **Log** dialog box that appears, click **View Logs** for **taskmanager.log** in the **Actions** column.



5. On the **Container Log** tab, view the log entries.



Note You can press **Ctrl+F** for Windows or **Cmd+F** for MacOS to search for specified log entries. We recommend that you view the logs from the last page. The first `error` log entry describes the root cause of the job error.

4.Machine Learning Platform for AI

4.1. What is machine learning?

Machine learning is a process of using statistical algorithms to learn large amounts of historical data and generate an empirical model to provide business strategies.

Apsara Stack Machine Learning Platform for AI is a set of data mining, modeling, and prediction tools. It is developed based on MaxCompute (also known as ODPS). Machine Learning Platform for AI supports the following functions:

- Provides an all-in-one algorithm service covering algorithm development, sharing, model training, deployment, and monitoring.
- Allows you to complete the entire procedure of an experiment either through the GUI or by running PAI commands. This function is typically intended for data mining personnel, analysts, algorithm developers, and data explorers.
- In Apsara Stack, Machine Learning Platform for AI runs on MaxCompute. Machine Learning Platform for AI allows you to call algorithms to decouple the applications and compute engines after you have deployed algorithm packages in MaxCompute clusters.
- Provides various algorithms and reliable technical support, providing more options to resolve service issues. In the Data Technology (DT) era, you can use Machine Learning Platform for AI to implement data-driven services.

Machine Learning Platform for AI can be applied in the following scenarios:

- Marketing: commodity recommendations, user profiling, and precise advertising.
- Finance: loan delivery prediction, financial risk control, stock trend prediction, and gold price prediction.
- Social network sites (SNSs): microblog leader analysis and social relationship chain analysis.
- Text: news classification, keyword extraction, text summarization, and text analysis.
- Unstructured data processing: image classification and image text extraction through OCR.
- Other prediction cases: rainfall forecast and football match result prediction.

Machine learning can be divided into three types:

- Supervised learning: Each sample has an expected value. You can create a model and map input feature vectors to target values. Typical examples of this learning mode include regression and classification.
- Unsupervised learning: No samples have a target value. This learning mode is used to discover potential regular patterns from data. Typical examples of this learning mode include simple clustering.
- Reinforcement learning: This learning mode is complex. A system constantly interacts with the external environment to obtain external feedback and determines its own behavior to achieve a long-term optimization of targets. Typical examples of this learning mode include AlphaGo and driverless vehicles.

4.2. Quick start

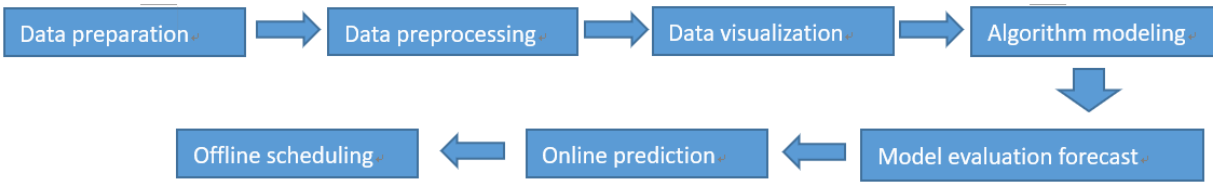
4.2.1. Overview

This topic describes how to perform data preparation, data preprocessing, data visualization, algorithm modeling, model prediction and evaluation, online prediction, and DataWorks task scheduling to set up a machine learning experiment.

> **Note** This document covers Apsara Stack Machine Learning Platform for AI, online model service, and deep learning framework. The online model service and deep learning framework are not basic functions of Apsara Stack Machine Learning Platform for AI and must be purchased separately.

For more information, see Machine learning experiment creation flowchart.

Machine learning experiment creation flowchart



1. Data preparation

   Import target data into the Apsara Stack Machine Learning Platform for AI console.

2. Data preprocessing

   Perform data processing, including SQL-based conversion, normalization, and standardization, to ensure that all data has the same dimensions.

3. Data visualization

   Display data in charts to view the features of the data and the distribution of the values. This serves as the basis for model algorithm selection.

4. Algorithm modeling

   Use machine learning algorithms to train data and ultimately build a model.

5. Model evaluation forecast

   Make predictions from and evaluate the model, and use the prediction results to create business development strategies.

6. Online prediction

   Use online prediction to deploy the generated model and adjust your business strategy based on the prediction results.

7. Offline scheduling

   Deploy experiments in DataStudio and run them on a regular basis.

4.2.2. Log on to Apsara Stack Machine Learning Platform for AI

This topic describes how to log on to Apsara Stack Machine Learning Platform for AI.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

Procedure

1. Enter the address into the address bar of your web browser, and then press Enter.
2. Enter your username and password. Obtain the username and password for logging on to the console from the operations administrator.

   > **Note** The first time you are logging on to ASCM, you must follow the instructions to change the password of your account. For security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain two or more types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to log on to ASCM.
4. In the top menu bar, choose **Products** > **Machine Learnig Platform for AI** to open the portal of Machine Learning Platform for AI.
5. Select an organization, and then click **PAI**.

   > **Note** If this is the first time that you log on to the Machine Learning Platform for AI console, you must perform the following tasks:
   >
   > i. Add an organization
   >
   > Create an organization to manage resource sets and resources in the resource sets.
   >
   > ii. Create a user
   >
   > Administrators can create users and attach different roles to users for permission control.
   >
   > iii. Create a resource set
   >
   > Create a resource set before you apply for resources.
   >
   > iv. Add a member to a resource set
   >
   > Add users to the resource set as members.
   >
   > v. In the top menu bar of ASCM, choose **Products** > **Big Data** > **MaxCompute**, and then create a **task account** and **MaxCompute project**.
   >
   > a. Create a task account: set the **Organization** to the one created in step 1.
   >
   > b. Create a MaxCompute project: set the **Organization** to the one created in step 1, set the **Resource Set** to the one created in step 2, and set the **Task Account** to the one that you have created.
   >
   > vi. Create a DataWorks workspace. In the **Advanced Settings** section, set **MaxCompute Project Name** to the project created in the preceding step.

4.2.3. Data preparation

This topic describes how to import data into the Apsara Stack Machine Learning Platform for AI console for modelling.

Prerequisites

Make sure that you have created a MaxCompute project and imported table data into the project. You can download the data from .

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console and click **Experiments** in the left-side navigation pane.
2. On the **Experiments** page, right-click **My Experiments** and choose **New Experiment** from the shortcut menu. In the dialog box that appears, enter the experiment name and description. Click **Create** to go to the **Components** page.
3. In the **Components** list, click **Data Source/Target**, and drag and drop the **Read MaxCompute Table** onto the canvas.
4. Click the **Read MaxCompute Table** component and configure its parameters. Enter the MaxCompute table name in **Table Name** in the right-side configuration pane.
5. In the right-side parameter setting pane, click **Column Information** to view the column name, data type, and the first 100 rows of data in the input table.

4.2.4. Data preprocessing

This topic describes how to perform data preprocessing by using methods such as normalization, SQL scripts, and data splitting.

Prerequisites

Before data preprocessing, make sure that you have completed data preparation.

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console and click **Components** in the left-side navigation pane.
2. In the **Components** list, click **Tools**. Drag and drop the **SQL Script** component onto the canvas. Click **Data Preprocessing**. Drag and drop the **Normalization** component onto the canvas, and connect the components.
3. Click the **SQL Script** component and click the right-side **Parameters** tab. In the **SQL Script** input box, enter the following SQL scripts to convert the features from string type to numeric type.

```
select age,
(case sex when 'male' then 1 else 0 end) as sex,
(case cp when 'angina' then 0  when 'notang' then 1 else 2 end) as cp,
trestbps,
chol,
(case fbs when 'true' then 1 else 0 end) as fbs,
(case restecg when 'norm' then 0  when 'abn' then 1 else 2 end) as restecg,
thalach,
(case exang when 'true' then 1 else 0 end) as exang,
oldpeak,
(case slop when 'up' then 0  when 'flat' then 1 else 2 end) as slop,
ca,
(case thal when 'norm' then 0  when 'fix' then 1 else 2 end) as thal,
(case status  when 'sick' then 1 else 0 end) as ifHealth
from  ${t2};
```

4. Click the **Normalization** component and select all fields to normalize the numeric features to values ranging from 0 to 1.
5. Click **Data Preprocessing**. Drag and drop the **Split** component onto the canvas and set **Split Ratio** to **0.7**.

> **Note** This step splits data into two parts: 70% of the data is used as the model training set, and 30% of the data is used as the model prediction set.

4.2.5. Data visualization

This topic describes how to view the features and value distribution by using statistical analysis components.

Prerequisites

Before data visualization, make sure that you have completed data preprocessing.

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console and click **Components** in the left-side navigation pane.
2. In the **Components** list, click **Statistical Analysis**. Drag and drop the **Whole Table Statistics** component onto the canvas. Connect the components, and click **Run** at the bottom of the canvas.
3. After the experiment stops running, right-click **Whole Table Statistics** and choose **View Data** from the shortcut menu. The analysis report is displayed.

4.2.6. Algorithm modeling

This topic describes how to perform feature training and generate models by using the machine learning components.

Prerequisites

Before algorithm modeling, ensure that you have completed data preprocessing and learned the data characteristics and value distribution through data visualization.

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console and click **Components** in the left-side navigation pane.
2. In the **Components** list, choose **Machine Learning** > **Binary Classification**. Drag and drop the **Binary Logistic Regression** component onto the canvas, and connect the corresponding components and data streams.
3. Click the component, and select 13 feature columns from **Training Feature Columns** in the right-side **Column Settings** pane. All parameters use the default settings.
4. Click **Run**.
5. Click **Models** in the left-side navigation pane to view the generated model.

4.2.7. Model prediction evaluation

This topic describes how to use a model to make predictions and evaluate its results by using the prediction and evaluation components.

Prerequisites

Before evaluating the prediction, make sure that you have completed algorithm modeling and generated a machine learning model from the experiment.

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console and click **Components** in the left-side navigation pane.
2. In the **Components** list, click **Machine Learning**. Drag and drop the **Prediction** component onto the canvas, and connect the corresponding components and data streams.
3. Choose **Machine Learning** > **Evaluation**. Drag and drop the **Binary Classification Evaluation** component onto the canvas and connect the corresponding components and data streams.
4. Click **Run** in the upper-left corner of the canvas. During the running process, select a component and click the **Developer Tool** ( ) icon in the lower-right corner of the canvas to view the running status of the component.
5. Right-click the **Binary Classification Evaluation** component and choose **View Evaluation Report** from the shortcut menu to generate the ROC curve of the LR model trained with different parameters.

4.2.8. DataWorks task scheduling

After you have run all nodes in an experiment, you can deploy the experiment to DataWorks and schedule DataWorks to periodically run the experiment. This topic uses air quality prediction as an example scenario.

Prerequisites

Before scheduling an experiment, you must make sure that you have successfully run all nodes in the experiment and that the experiment is deployed to DataWorks.

Procedure

1. Log on to the Apsara Stack Machine Learning Platform for AI console. Click **Experiments** in the left-side navigation pane.
2. Click the **My Experiments** tab and select an experiment to navigate to the canvas.

   > **Notice** Make sure all components have been run in the experiment. A green check means that the component is running correctly.

3. In the upper-left corner of the canvas, choose **Deploy > Schedule DataWorks Tasks** to go to DataStudio.
4. In the DataStudio console, choose **Create > Algorithm > Machine Learning Platform for AI**, and then create a Machine Learning experiment node.
5. In the **Create Node** dialog box, enter the node name, select the target folder, and click **Submit**.

   > **Notice** You must select a target folder for the algorithm type.

   After the experiment node is created.
6. Select the experiment from the drop-down list.
7. Configure task scheduling parameters, including the recurrence, input, and output parameters.
8. Click **Submit**. The task will be executed the next day.
9. Click **Administration** in the upper-right corner to go to the administration page. You can view the status of the machine learning task and the system log. You can also perform other operations such as adding retroactive data and testing the experiment.

4.3. Online model service (must be activated separately)

4.3.1. Deploy an online model service

This topic describes how to deploy the generated experiment model through the online model service. You can adjust your business strategy anytime based on predicted results.

Prerequisites

Before deploying the online model service, make sure that the preceding steps are completed and the components are running properly. A green check means that the component is running

correctly.

Procedure

1. Log on to the Apsara Stack Machine Learning for AI console and click **Experiments** in the left-side navigation pane.
2. Click the **My Experiments** tab and select an experiment to navigate to the canvas.

   > **Notice** Make sure that the selected experiment is running properly. A green check means that the component is running correctly.

3. In the upper-left corner of the canvas, choose **Deploy** > **Online Model Service**.
4. Select the model to deploy and click **Next**.
5. Select a deployment mode. You can select one of the following modes:
   - New Service
   - Add Existing Service Version
   - Create Blue-green Deployment

4.3.2. Create a service

This topic describes how to use the **New Service** mode to deploy online prediction services.

Procedure

1. Complete Preparations for online model prediction.
2. Set **Processes** and **Quota**.

   > **Note** **Processes** determines the maximum number of concurrently running programs. **Quota** determines the running speed and the parameters such as RT and QPS.

3. Click **Deploy**. It takes several minutes to create the model.
4. After the model is created, click the model name to view information about the model invocation.
5. Click the icons under **Monitor** to view statistics about QPS, response, RT, traffic, CPU utilization, memory usage, and daily invocation.
6. > **Note** Perform this step when resources are insufficient and need to be expanded.

   Click **Update** to expand resources.
7. Click **Online Debugging** in the upper-right corner of the page and select the current model.

4.3.3. Add an existing service version

This topic describes how to use the **Add Existing Service Version** mode to deploy online prediction services.

Prerequisites

Before you use the **Add Existing Service Version** mode to deploy online prediction services, ensure that you have deployed one version of online prediction services through the New Service mode.

Procedure

1. Complete Preparations for online model prediction.
2. Select **Add Existing Service Version**.
3. Select a deployed model. It takes several minutes to add a version.
4. After the model is deployed, select the added version from the **Current Version** drop-down list.

4.3.4. Create a blue-green deployment

This topic describes how to use the **Create Blue-green Deployment** mode to deploy online prediction services.

Prerequisites

Before you use the **Create Blue-green Deployment** mode to deploy online prediction services, ensure that you have deployed two versions of online prediction services through the New Service and Add Existing Service Version modes.

Context

In the blue-green deployment mode, you can deploy and test the target version without stopping the source version. After confirming that the target version is running normally, switch all traffic to the target version. Blue-green deployment is safe and does not interrupt services.

Procedure

1. Complete Preparations for online model prediction.
2. Select the deployed model service and click **Deploy**. The deployment may take several minutes.
3. Click **Switch Traffic** and adjust the ratio of traffic forwarded to the two models. The initial ratio is 100% for both models.
4. Perform online debugging.
   i. Click **Online Debugging** in the upper-right corner of the page and select the deployed model.
   ii. Enter data (feature input) in **Body**. For example, the body information of the logistic regression model for heart disease prediction is as follows:

   ```
   [{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"th
   alach":0,"oldpeak":0,"ca":0}]
   ```

   iii. Click **Run** and check the result.

4.4. Components

4.4.1. Overview

This topic describes how to use and configure machine learning components. When building a machine learning experiment, you can select components based on the features of existing data to generate a model and make accurate predictions for your business.

Each component has one or more input or output ports. You can move the pointer over the ports to view their descriptions and connect the components.

4.4.2. Data source and target

This topic describes components in the **Data Source/Target** category, such as the Read MaxCompute Table and Write MaxCompute Table components.

Read MaxCompute tables

You can use the Read MaxCompute Table component to read MaxCompute tables. By default, this component reads data of the current project. If you want to read data from tables in another project for which you have access, you can prefix the table name with the project name in the `project name. table name` format. For example, `tianchi_project.weibo_data`. After you specify the input table, the system reads the structural data of the table. You can click the **Column Information** tab to view the data. This component does not support views.

If the selected input table is a partitioned table, the back end automatically selects the Partition checkbox. You can select or configure partition parameters. Only one partition can be selected. If you do not select the Partition checkbox or do not specify the partition parameters, the whole table is selected. If the input table is non-partitioned, the Partition checkbox cannot be selected.

Write MaxCompute tables

You can use the Write MaxCompute Table component to write data to tables in the current project or tables in other projects. This component can write data to partitions. Partitions must be created for the table in the MaxCompute console before this component can write data to the partitions. You can set the table lifecycle measured in days.

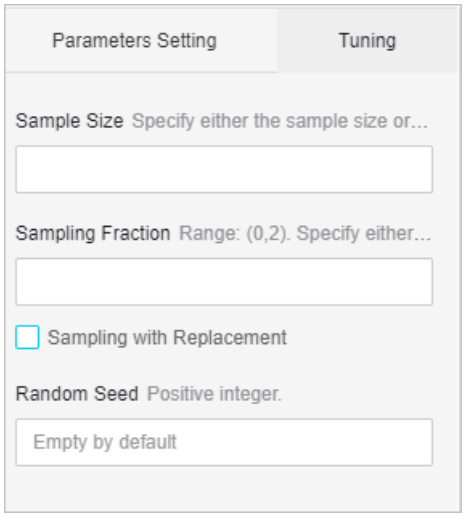
4.4.3. Data preprocessing

4.4.3.1. Sampling and filtering

4.4.3.1.1. Random sampling

Data is sampled randomly and independently. You can specify a ratio or quantity of samples to be taken and choose whether to enable sampling with replacement.

Parameter settings



PAI command

```
Pai -name sample
-project algo_public
-DinputTableName=wbpc
-DoutputTableName=wpbc_sample
-Dratio=0.3;
```

Algorithm parameters

Parameters

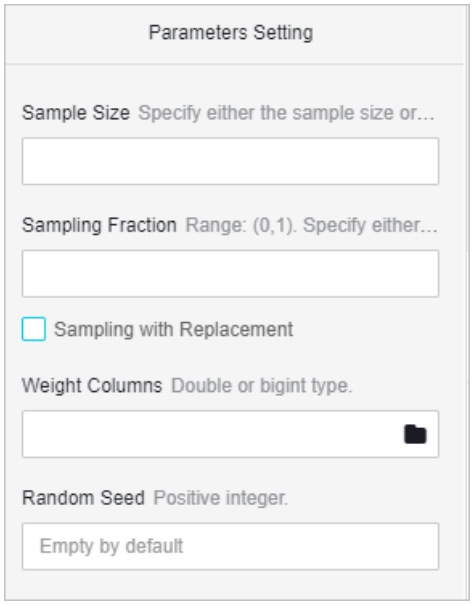
Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.

Parameter	Description	Valid values	Default value
ratio	Required. The sampling fraction.	(0, 1)	-
outputTableName	Required. The name of the output table.	-	-
outputTablePartition	Optional. The partition of the output table.	-	The output table is a non-partitioned table by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer in the range of [1, 3650]	No lifecycle is set by default.

4.4.3.1.2. Weighted sampling

Sample data is collected based on weights. The weight column must be of double or int type. Data is sampled based on the value of its corresponding weight. For example, data with a col value of 1.2 has a higher probability to be sampled than data with a col value of 1.0.

Parameter settings



Parameter settings

Parameter	Description
Sample Size	You can specify the number of samples to be taken, which is 10,000 by default. For sampling without replacement, the number of samples cannot be greater than the number of data entries.
Sampling Fraction	You can use either the Sample Size or Sampling Fraction parameter. You can choose sampling with or without replacement, the latter of which is used by default. Select the checkbox to enable sampling with replacement.

Parameter	Description
Weight Columns	You can select a weight column from the drop-down list. The weight column can be of the double or bigint type.
Random Seed	The random seed, which is a positive integer. This parameter is empty by default.

- You can choose sampling with or without replacement, the latter of which is used by default. Select the checkbox to enable sampling with replacement.
- You can specify the number of samples to be taken, which is 10,000 by default.

  > **Note** For sampling without replacement, the number of samples cannot be greater than the number of data entries.

- You can select a weight column from the drop-down list. The weight column can be of the double or bigint type.

PAI command

```
PAI -name WeightedSample
-project algo_public
-DprobCol="previous"
-DsampleSize="500"
-DoutputTableName="test2"
-DinputPartitions="pt=20150501"
-DinputTableName="bank_data_partition";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
replace	Indicates whether sampled data is replaced. If this parameter is set to true, data is replaced after it is sampled. If this parameter is set to false, data is not replaced after it is sampled.
probCol	The columns to be weighted. Each value indicates the weight of an entry. Normalization is not required.
sampleSize	The number of samples to be taken. For sampling without replacement, the number of samples cannot be greater than the number of data entries.

Parameter	Description
outputTableNames	The name of the output table. Separate multiple table names with commas (,).
inputPartitions	Optional. The partitions selected from the input table for training. If no partitions are specified, the entire table is selected.
inputTableName	The name of the input table.
replace	Optional. This parameter indicates whether sampled data is replaced. If this parameter is set to true, data is replaced after it is sampled. If this parameter is set to false, data is not replaced after it is sampled.

4.4.3.1.3. Filtering and mapping

You can filter data based on filtering expressions and rename columns.

Parameter settings

1. Use the WHERE condition to filter data similar to how it would function in an SQL statement.

   Filtering conditions: Operators available include the equal (=), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=) signs, as well as like and rlike.

2. Rename columns.

PAI command

```
PAI -name Filter
-project algo_public
-DoutTableName="test_9"
-DinputPartitions="pt=20150501"
-DinputTableName="bank_data_partition"
-Dfilter="age>=40";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
outTableName	The name of the output table.
inputPartitions	Optional. The partitions selected from the input table for training. If no partitions are specified, the entire table is selected.

Parameter	Description
inputTableName	The name of the input table.
filter	The WHERE condition to filter data. Operators available include the equal (=), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=) signs, as well as like and rlike.

4.4.3.1.4. Stratified sampling

Stratified sampling is a statistical computing method. It works by dividing a population into several strata based on specified features, performing random sampling at each stratum, and creating a sample collection.

Parameter settings

Parameter	Description
Column Settings	Stratification Column: Required. Samples are stratified based on this column.
Parameter Settings	Sampling Fraction/Sample Size: Required. A value less than 1 represents the sampling fraction per stratum. A value greater than 1 represents the number of samples at each stratum.
	Other Sampling Configurations: Optional. This parameter allows you to collect different numbers of samples at different strata.
	Random Seed: Optional. Valid values: 1, 2, 3, 4, 5, 6, and 7.

PAI command

```
Pai -name sample
-project algo_public
-DinputTableName=wbpc
-DoutputTableName=wpbc_sample
-DstrataColName="label"
-DsampleSize="A:200,B:300,C:500"
-DrandomSeed=1007
-Dlifecycle=30
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.
strataColName	Required. The stratification column.	-	-
outputTableName	Required. The name of the output table.	-	-
sampleSize	Optional. An integer value that specifies the number of samples taken from each stratum. A string value must be in the `strata0:n0,strata1:n1....` format. Each item in the string represents the number of samples to be taken from the corresponding stratum.	-	

Parameter	Description	Valid values	Default value
sampleRatio	Optional. A decimal value from 0 to 1 that represents the ratio of data for each stratum to be sampled. A string value must be in the `strata0:r0,strata1:r1,...` format. Each item in the string represents the sampling fraction for the corresponding stratum.	-	-
randomSeed	Optional. The number of random seeds.	-	0
lifecycle	Optional. The lifecycle of the output table.	A positive integer in the range of [1, 3650]	No lifecycle is set by default.
coreNum	Optional. The number of cores.	-	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	-	Automatically calculated.

4.4.3.2. Data merge

4.4.3.2.1. Join

This component merges two tables by associating the information in the tables and outputting the specified columns. This component is similar to the JOIN statement of SQL.

Parameter settings

- Join types: left join, internal join, right join, and full join.
- Only the equation condition is supported.
- You can manually add or delete join conditions.

PAI command

No PAI command is available.

4.4.3.2.2. Merge columns

You can merge data of two tables by column. The two tables must have the same number of rows.

Parameter settings

Procedure

1. Select input columns from the left table.
2. Select input columns from the right table.

   When merging columns:

   - The two tables must have the same number of rows.
   - The names of output columns selected from the left and right tables cannot be the same.
   - When selecting an output column, you can change its name.
   - If no output columns are selected from the left or right table, the whole table is selected. In this case, if **Automatically Rename Output Columns** is selected, the duplicate columns are renamed and then output.

PAI command

```
PAI -name AppendColumns
-project algo_public
-DoutputTableColNames="petal_length,petal_width,petal_length2,petal_width2"
-DautoRenameCol="false"
-DoutputTableName="pai_temp_770_6840_1"
-DinputTableNames="iris_twopartition,iris_twopartition"
-DinputPartitionsInfoList="dt=20150125/dp=20150124;dt=20150124/dp=20150123"
-DselectedColNamesList="petal_length,petal_width;sepal_length,sepal_width";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
outputTableColNames	The names of the columns in the new table. The column names must be separated with commas (,). If autoRenameCol is set to true, this parameter is ignored.
autoRenameCol	Optional. This parameter specifies whether to automatically rename the columns in the output table. If the value is true, the columns are renamed. If the value is false, the columns are not renamed. Default value: false.
outputTableName	The name of the output table.
inputTableNames	The name of the input table. Separate multiple table names with commas (,).

Parameter	Description
inputPartitionsInfoList	Optional. A list of partitions selected from the corresponding input tables. Partitions of the same table must be separated with commas (,) and partitions of different tables must be separated with semicolons (;).
selectedColNamesList	The names of selected columns. The names of columns in the same table must be separated with commas (,) and the names of columns in different tables must be separated with semicolons (;).

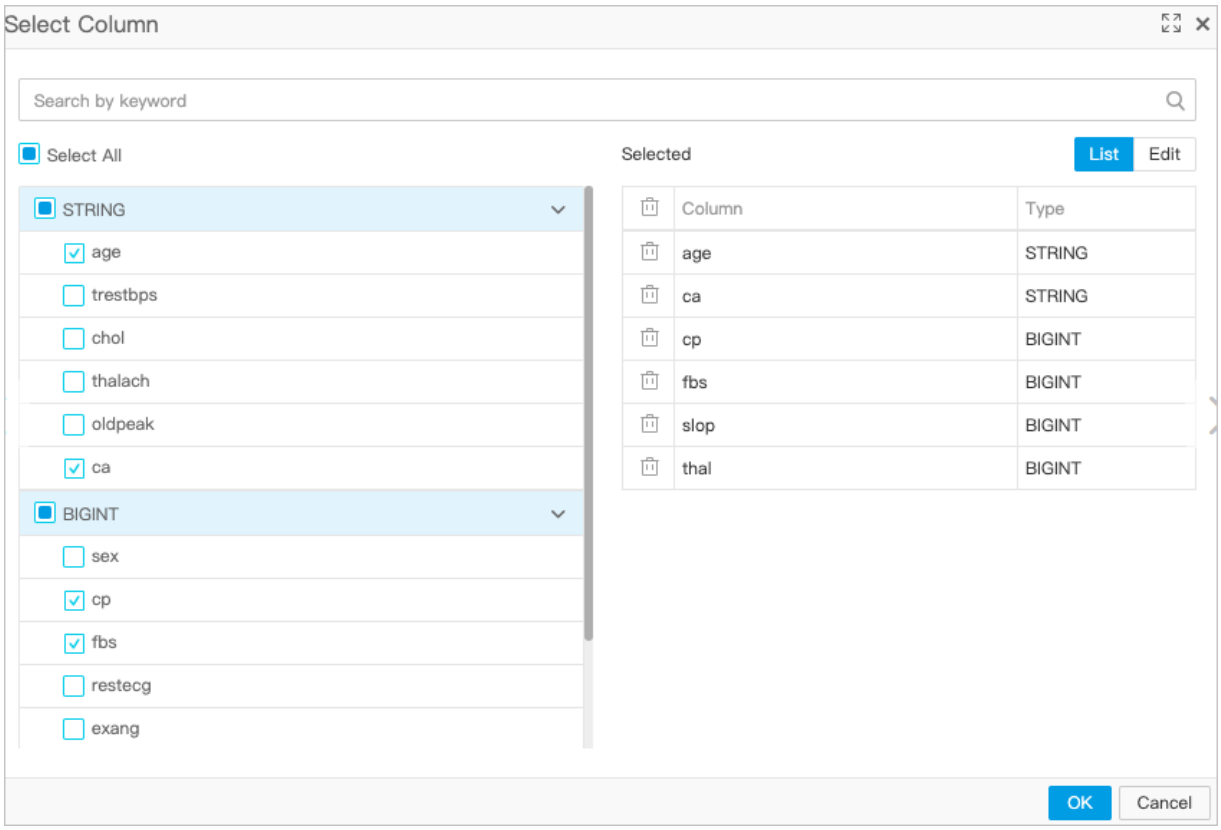
4.4.3.2.3. Merge rows (UNION)

To merge the data of two tables by row, the quantity and data type of the output columns selected from the left and right tables must be the same. The function is integrated with the UNION and UNION ALL functions.

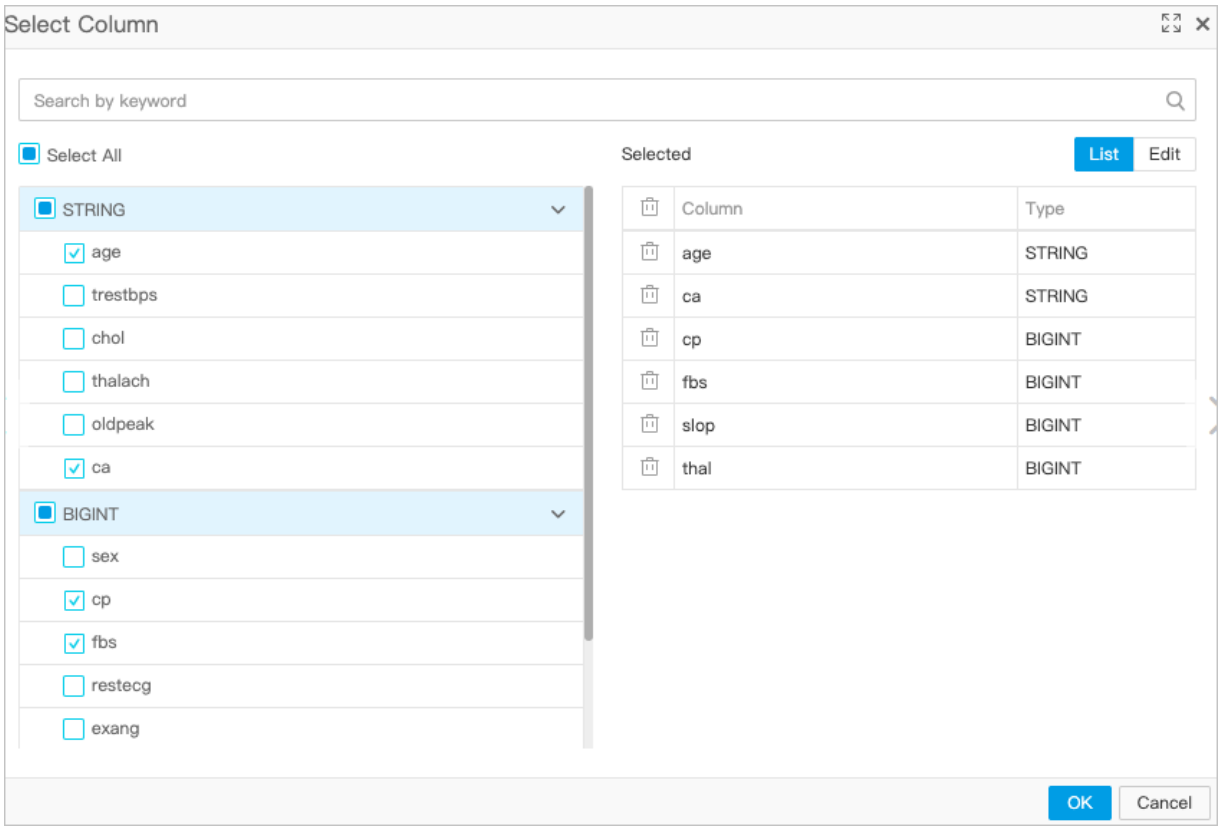
Parameter settings

- During the merge process, the numbers of columns selected from the left and right tables must be the same, and the data types of the corresponding columns must be the same.
- You can enter conditions in the text box by which to filter and select columns. The whole table is selected by default. Operators available include the equal (=), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=) signs, as well as like and rlike.
- **Remove Duplicates** is selected by default. When this option is selected, duplicate rows in the output table are removed.

The following figure shows the union columns selected from the left table.



The following figure shows the union columns selected from the right table.



PAI command

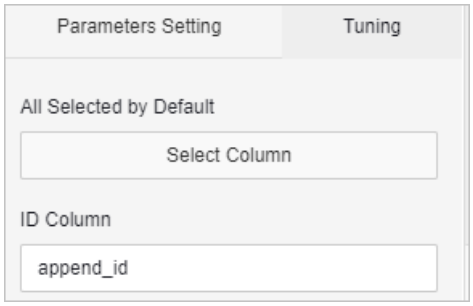
No PAI command is available.

4.4.3.3. Others

4.4.3.3.1. Add ID column

You can append an ID column to a table as the first column and save the table as a new table.

Parameter settings



PAI command

```
PAI -name AppendId
-project algo_public
-DIDColName="append_id"
-DoutputTableName="test_11"
-DinputTableName="bank_data"
-DselectedColNames="age,campaign,cons_conf_idx,cons_price_idx,emp_var_rate,euribor3m,nr_employ
ed,pdays,poutcome,previous,y";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
IDColName	The name of the appended ID column. ID numbers start from 0 and increment by one. Example: 0, 1, 2, 3, ...
outputTableName	The name of the output table.
inputTableName	The name of the input table.

Parameter	Description
selectedColNames	The names of the columns to be retained. Separate multiple columns with commas (,).

4.4.3.3.2. Split

This component is used to split an input table or a partition based on a specified ratio, and output two tables from two output ports.

Algorithm component

Parameter settings

- The Split component has two output ports.
- In Parameter settings, if the splitting fraction is set to 0.7, the left output port outputs 70% of the data and the right output port outputs 30% of the data.

PAI command

```
pai -name split -project algo_public
    -DinputTableName=wbpc
      -Doutput1TableName=wpbc_split1
    -Doutput2TableName=wpbc_split2
    -Dfraction=0.25;
```

Parameter settings

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.
output1TableName	Required. The name of output table 1.	-	-

Parameter	Description	Valid values	Default value
output1TablePartition	Optional. The partitions in output table 1.	-	Output table 1 is a non-partitioned table by default.
output2TableName	Required. The name of output table 2.	-	-
output2TablePartition	Optional. The partitions in output table 2.	-	Output table 2 is a non-partitioned table by default.
fraction	Required. The portion of data diverted to output table 1.	(0, 1)	-
lifecycle	Optional. The lifecycle of the output table.	A positive integer in the range of [1, 3650]	No lifecycle is set by default.

4.4.3.3.3. Missing value imputation

This component replaces a null value or a specified value with the maximum, minimum, average, or custom value. A list of values is defined to impute the missing values in an input table with the specified values.

- This component can replace a numeric null value with the maximum, minimum, average, or custom value.
- This component can also replace a null string, empty string, null and empty string, or specified value with a custom value.
- The missing values to be imputed can be null strings, empty strings, or custom values. If you choose empty strings, the data type of the target column must be string.

The parameters for the two input ports are as follows:

- inputTableName: the name of the input table for which to replace missing data.
- inputParaTableName: the name of the input configuration table that contains parameters generated by the missing value imputation node. Based on this parameter, configuration parameters in one table can be applied to a new table.

Parameters for the two output ports are as follows:

- outputTableName: the name of the imputed output table.
- outputParaTableName: the name of the output configuration table, which can be applied to other datasets.
- Columns to Impute: the names of the columns for which to replace missing values.
- Original Value: the values to be replaced.
- Replaced With: the replacement values.

PAI command

```
PAI -name FillMissingValues
    -project algo_public
    -Dconfigs="poutcome,null-empty,testing" \
    -DoutputTableName="test_3"
    -DinputPartitions="pt=20150501"
    -DinputTableName="bank_data_partition";
```

Algorithm parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	Table name	N/A
inputTablePartitions	Optional. The partitions selected from the input table for training.	Partition name	The whole table is selected by default.
outputTableName	Required. The name of the output table.	Table name	N/A

Parameter	Description	Valid value	Default value
configs	Required. The configurations for missing value imputation. Example: `col1, null, 3.14; col2, empty, hello; col3, empty-null, world`, where *null* indicates a null value and *empty* indicates an empty string. If you choose to use empty strings to fill the target columns, the data type of the target column must be string. The variables used to specify the replacement value as maximum, minimum, and average are max, min, and mean respectively. If you want to impute a custom value to the target column, use a user-defined variable in the `col4,user-defined,str,str123` format.	N/A	N/A
outputParaTableName	Required. The name of the output configuration table.	Table name	N/A
inputParaTableName	Optional. The name of the input configuration table.	Table name	No input configuration table is set by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	A positive integer	Automatically calculated.

Examples

Test data

- SQL statement to generate data:

```
drop table if exists fill_missing_values_test_input;
create table fill_missing_values_test_input(
  col_string string,
  col_bigint bigint,
  col_double double,
  col_boolean boolean,
  col_datetime datetime);
insert overwrite table fill_missing_values_test_input
select
  *
from
(
  select
    '01' as col_string,
    10 as col_bigint,
    10.1 as col_double,
    True as col_boolean,
    cast('2016-07-01 10:00:00' as datetime) as col_datetime
  from dual
  union all
    select
      cast(null as string) as col_string,
      11 as col_bigint,
      10.2 as col_double,
      False as col_boolean,
      cast('2016-07-02 10:00:00' as datetime) as col_datetime
    from dual
  union all
    select
      '02' as col_string,
      cast(null as bigint) as col_bigint,
      10.3 as col_double,
      True as col_boolean,
      cast('2016-07-03 10:00:00' as datetime) as col_datetime
    from dual
  union all
    select
      '03' as col_string,
      12 as col_bigint,
```

```
        cast(null as double) as col_double,
        False as col_boolean,
        cast('2016-07-04 10:00:00' as datetime) as col_datetime
    from dual
  union all
    select
      '04' as col_string,
      13 as col_bigint,
      10.4 as col_double,
      cast(null as boolean) as col_boolean,
      cast('2016-07-05 10:00:00' as datetime) as col_datetime
    from dual
  union all
    select
      '05' as col_string,
      14 as col_bigint,
      10.5 as col_double,
      True as col_boolean,
      cast(null as datetime) as col_datetime
    from dual
) tmp;
```

- Input description

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04         | 13         | 10.4       | NULL        | 2016-07-05 10:00:00 |
| 02         | NULL       | 10.3       | true        | 2016-07-03 10:00:00 |
| 03         | 12         | NULL       | false       | 2016-07-04 10:00:00 |
| NULL       | 11         | 10.2       | false       | 2016-07-02 10:00:00 |
| 01         | 10         | 10.1       | true        | 2016-07-01 10:00:00 |
| 05         | 14         | 10.5       | true        | NULL         |
+------------+------------+------------+-------------+--------------+
```

PAI command

```
drop table if exists fill_missing_values_test_input_output;
drop table if exists fill_missing_values_test_input_model_output;
PAI -name FillMissingValues
-project algo_public
-Dconfigs="col_double,null,mean;col_string,null-empty,str_type_empty;col_bigint,null,max;col_boolean,
null,true;col_datetime,null,2016-07-06 10:00:00"
-DoutputParaTableName="fill_missing_values_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="fill_missing_values_test_input_output"
-DinputTableName="fill_missing_values_test_input";
drop table if exists fill_missing_values_test_input_output_using_model;
drop table if exists fill_missing_values_test_input_output_using_model_model_output;
PAI -name FillMissingValues
-project algo_public
-DoutputParaTableName="fill_missing_values_test_input_output_using_model_model_output"
-DinputParaTableName="fill_missing_values_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="fill_missing_values_test_input_output_using_model"
-DinputTableName="fill_missing_values_test_input";
```

Output

- fill_missing_values_test_input_output

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04         | 13         | 10.4       | true        | 2016-07-05 10:00:00 |
| 02         | 14         | 10.3       | true        | 2016-07-03 10:00:00 |
| 03         | 12         | 10.3       | false       | 2016-07-04 10:00:00 |
| str_type_empty | 11         | 10.2       | false       | 2016-07-02 10:00:00 |
| 01         | 10         | 10.1       | true        | 2016-07-01 10:00:00 |
| 05         | 14         | 10.5       | true        | 2016-07-06 10:00:00 |
+------------+------------+------------+-------------+--------------+
```

- fill_missing_values_test_input_model_output

```
+------------+------------+
| feature    | json       |
+------------+------------+
| col_string | {"name": "fillMissingValues", "type": "string", "paras":{"missing_value_type": "null-empt
y",  "replaced_value": "str_type_empty"}} |
| col_bigint | {"name": "fillMissingValues", "type": "bigint", "paras":{"missing_value_type": "null",  "rep
laced_value": 14}} |
| col_double | {"name": "fillMissingValues", "type": "double", "paras":{"missing_value_type": "null",  "r
eplaced_value": 10.3}} |
| col_boolean | {"name": "fillMissingValues", "type": "boolean", "paras":{"missing_value_type": "null",
"replaced_value": 1}} |
| col_datetime | {"name": "fillMissingValues", "type": "datetime", "paras":{"missing_value_type": "null
",  "replaced_value": 1467770400000}} |
+------------+------------+
```

- fill_missing_values_test_input_output_using_model

```
+------------+------------+------------+-------------+--------------+
| col_string | col_bigint | col_double | col_boolean | col_datetime |
+------------+------------+------------+-------------+--------------+
| 04         | 13         | 10.4       | true        | 2016-07-05 10:00:00 |
| 02         | 14         | 10.3       | true        | 2016-07-03 10:00:00 |
| 03         | 12         | 10.3       | false       | 2016-07-04 10:00:00 |
| str_type_empty | 11         | 10.2       | false       | 2016-07-02 10:00:00 |
| 01         | 10         | 10.1       | true        | 2016-07-01 10:00:00 |
| 05         | 14         | 10.5       | true        | 2016-07-06 10:00:00 |
+------------+------------+------------+-------------+--------------+
```

- fill_missing_values_test_input_output_using_model_model_output

```
+------------+------------+
| feature    | json       |
+------------+------------+
| col_string | {"name": "fillMissingValues", "type": "string", "paras":{"missing_value_type": "null-empty",  "replaced_value": "str_type_empty"}} |
| col_bigint | {"name": "fillMissingValues", "type": "bigint", "paras":{"missing_value_type": "null",  "replaced_value": 14}} |
| col_double | {"name": "fillMissingValues", "type": "double", "paras":{"missing_value_type": "null",  "replaced_value": 10.3}} |
| col_boolean | {"name": "fillMissingValues", "type": "boolean", "paras":{"missing_value_type": "null", "replaced_value": 1}} |
| col_datetime | {"name": "fillMissingValues", "type": "datetime", "paras":{"missing_value_type": "null",  "replaced_value": 1467770400000}} |
+------------+------------+
```

4.4.3.3.4. Normalization

You can normalize one or more columns in a table and save the generated data to a new table.

Linear function transformation is supported. The transformation expression is `y=(x-MinValue)/(MaxValue-MinValue)`.

MaxValue and *MinValue* indicate the maximum and minimum values of the sample respectively.

- Click **Columns** to select the columns to be normalized. Double and bigint types are supported.
- You can choose whether to retain the original columns. If you select the corresponding checkbox, the original columns will be retained. Processed columns will be renamed.

PAI command

```
PAI -name normalize_wf
-project algo_public
-DkeepOriginal="true"
-DoutputTableName="test_4"
-DinputPartitions="pt=20150501"
-DinputTableName="bank_data_partition"
-DselectedColNames="emp_var_rate,euribor3m";
```

Algorithm parameters

Parameters

Parameter	Description	Default value

Parameter	Description	Default value
inputTableName	Required. The name of the input table.	N/A
selectedColNames	Optional. The names of columns selected from the input table.	All columns are selected by default.
inputTablePartitions	Optional. The partitions selected from the input table for training.	The whole table is selected by default.
outputTableName	Required. The name of the output table.	N/A
outputParaTableName	Required. The name of the output configuration table.	N/A
inputParaTableName	Optional. The name of the input configuration table.	No input configuration table is set by default.
outputPMMLTableName	Required. The name of the output PMML table.	N/A
keepOriginal	Optional. This parameter specifies whether to retain the original columns. If `keepOriginal` is set to true, processed columns are renamed with the `normalized_` prefix and the original columns are retained and their data overwritten. If `keepOriginal` is set to false, all columns are retained but not renamed.	false
lifecycle	Optional. The lifecycle of the output table.	No lifecycle is set by default.
coreNum	Optional. The number of cores.	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	Automatically calculated.

4.4.3.3.5. Standardization

You can standardize one or more columns in a table and save the generated data to a new table.

- The formula used for standardization is `(X - Mean)/(Standard deviation)` .
  - Mean: The mean of samples.

- Standard deviation: The standard deviation of samples. This variable is used when samples are used to calculate the total deviation. To make the calculated value closer to the mean, you must moderately increase the calculated standard deviation by using the formula $\frac{1}{N-1}$.
- The formula for calculating the standard deviation of samples:

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}$$

$\bar{X}$ represents the mean of samples X1, X2, ..., Xn.

- You can choose whether to retain the original columns. If you select the corresponding checkbox, the original columns will be retained. Processed columns will be renamed.
- Click **Columns** and select columns to be standardized. Double and bigint types are supported.

PAI command

```
PAI -name Standardize
-project algo_public
-DkeepOriginal="false"
-DoutputTableName="test_5"
-DinputTablePartitions="pt=20150501"
-DinputTableName="bank_data_partition"
-DselectedColNames="euribor3m,pdays"
```

Standardization component

Parameters for the two input ports are as follows:

- inputTableName: the name of the input table to be standardized.
- inputParaTableName: the name of the input configuration table that contains the parameters generated by the standardization node. You can use an input configuration table to apply the configuration parameters of one table to a new table.

Parameters for the two output ports are as follows:

- outputTableName: the name of the standardized output table.
- outputParaTableName: the name of the output parameter table, which can be applied to other datasets.

Standardization parameters

The corresponding algorithm parameter for Reserve Original Columns is `keepOriginal`.

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
selectedColNames	Optional. The names of columns selected from the input table.	-	All columns are selected by default.
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.
outputTableName	Required. The name of the output table.	-	-
outputParaTableName	Required. The name of the output configuration table.	-	-
outputPartition	Optional. The partitions in the output table.	-	-
inputParaTableName	Optional. The name of the input configuration table.	-	No input configuration table is set by default.
keepOriginal	Optional. This parameter specifies whether to retain the original columns. If this parameter is set to true, the original columns are retained and the column name is suffixed with `_orig`.	true and false	false
lifecycle	Optional. The lifecycle of the output table.	-	No lifecycle is set by default.
coreNum	Optional. The number of cores.	-	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	-	Automatically calculated.

Examples

```
drop table if exists standardize_test_input;
create table standardize_test_input(
col_string string,
col_bigint bigint,
col_double double,
col_boolean boolean,
col_datetime datetime);
insert overwrite table standardize_test_input  select     * from  (
select       '01' as col_string,
10 as col_bigint,
10.1 as col_double,
True as col_boolean,
cast('2016-07-01 10:00:00' as datetime) as col_datetime    from dual    union all
select          cast(null as string) as col_string,
11 as col_bigint,
10.2 as col_double,
False as col_boolean,
cast('2016-07-02 10:00:00' as datetime) as col_datetime       from dual    union all
select
'02' as col_string,
cast(null as bigint) as col_bigint,
10.3 as col_double,
True as col_boolean,
cast('2016-07-03 10:00:00' as datetime) as col_datetime       from dual    union all
select          '03' as col_string,
12 as col_bigint,
cast(null as double) as col_double,
False as col_boolean,
cast('2016-07-04 10:00:00' as datetime) as col_datetime       from dual    union all
select          '04' as col_string,
13 as col_bigint,
10.4 as col_double,
cast(null as boolean) as col_boolean,
cast('2016-07-05 10:00:00' as datetime) as col_datetime       from dual    union all
select          '05' as col_string,
14 as col_bigint,
10.5 as col_double,
True as col_boolean,
cast(null as datetime) as col_datetime       from dual ) tmp;
```

PAI command

```
drop table if exists standardize_test_input_output;
drop table if exists standardize_test_input_model_output;
PAI -name Standardize
-project algo_public
-DoutputParaTableName="standardize_test_input_model_output"
-Dlifecycle="28"
-DoutputTableName="standardize_test_input_output"
-DinputTableName="standardize_test_input"
-DselectedColNames="col_double,col_bigint"
-DkeepOriginal="true";
drop table if exists standardize_test_input_output_using_model;
drop table if exists standardize_test_input_output_using_model_model_output;
PAI -name Standardize
-project algo_public
-DoutputParaTableName="standardize_test_input_output_using_model_model_output"
-DinputParaTableName="standardize_test_input_model_output"
-Dlifecycle="28"
 -DoutputTableName="standardize_test_input_output_using_model"
-DinputTableName="standardize_test_input"
```

Input description

standardize_test_input

col_string	col_bigint	col_double	col_boolean	col_datetime
01	10	10.1	true	2016-07-01 10:00:00
NULL	11	10.2	false	2016-07-02 10:00:00
02	NULL	10.3	true	2016-07-03 10:00:00
03	12	NULL	false	2016-07-04 10:00:00
04	13	10.4	NULL	2016-07-05 10:00:00
05	14	10.5	true	NULL

Output description

standardize_test_input_output

col_string	col_bigint	col_double	col_boolean	col_datetime	stdized_col_bigint	stdized_col_double
01	10	10.1	true	2016-0	-1.264911064	-1.264911064
NULL	11	10.2	false	2016-07-02 10:00:00	-0.6324555320336759	-0.6324555320341972
02	NULL	10.3	true	2016-07-03 10:00:00	NULL	0.0
03	12	NULL	false	2016-07-04 10:00:00	0.0	NULL
04	13	10.4	NULL	2016-07-05 10:00:00	0.6324555320336759	0.6324555320341859
05	14	10.5	true	NULL	1.2649110640673518	1.2649110640683718

standardize_test_input_model_output

Feature	json
col_bigint	{"name": "standardize", "type":"bigint", "paras": {"mean":12, "std": 1.58113883008419}}
col_double	{"name": "standardize", "type":"double", "paras": {"mean":10.3, "std": 0.1581138830082909}}

standardize_test_input_output_using_model

col_string	col_bigint	col_double	col_boolean	col_datetime
01	-1.2649110640673515	-1.2649110640683833	true	2016-07-01 10:00:00
NULL	-0.6324555320336758	-0.6324555320341971	false	2016-07-02 10:00:00
02	NULL	0.0	true	2016-07-03 10:00:00
03	0.0	NULL	false	2016-07-04 10:00:00

col_string	col_bigint	col_double	col_boolean	col_datetime
04	0.6324555320336758	0.6324555320341858	NULL	2016-07-05 10:00:00
05	1.2649110640673515	1.2649110640683716	true	NULL

standardize_test_input_output_using_model_model_output

feature	json
col_bigint	{"name": "standardize", "type":"bigint", "paras": {"mean":12, "std": 1.58113883008419}}
col_double	{"name": "standardize", "type":"double", "paras": {"mean":10.3, "std": 0.1581138830082909}}

4.4.3.3.6. KV to Table

This component is used to convert KV pairs to a table. The key is converted to a table column, while the value is converted to a column value in the corresponding row.

KV table format definition:

- A key is the index of a column. Key values can be of the bigint or double types.
- A KV table can be input in the sparse format to algorithm components such as logistic and linear regression.
- Keys must be of the string type. You can input a key_map table to the KV to Table component to map keys to columns. This component outputs a key_map table that contains all key-column mappings after conversion, regardless of whether you input a key_map table.

kv
1:10;2:20;3:30

key_map table format definition: a table that contains index-to-column mappings and data type information. The data types of the col_name, col_index, and col_datatype columns must be string. The default data type of the col_datatype column is double if not specified.

col_name	col_index	col_datatype
col1	1	bigint
col2	2	double

PAI command

```
PAI -name KVToTable
    -project algo_public
    -DinputTableName=test
    -DoutputTableName=test_out
    -DoutputKeyMapTableName=test_keymap_out
    -DkvColName=kv;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	The table cannot be empty.
kvColName	Required. The name of the KV column.	Only one column can be selected.	-
outputTableName	Required. The name of the output table.	Table name	-
outputKeyMapTableName	Required. The name of the output index table.	Table name	-
inputKeyMapTableName	Optional. The name of the input index table.	Table name	No input index table is set by default.
appendColName	Optional. The name of the appended column.	Multiple columns can be selected.	No column is appended by default.
inputTablePartitions	Optional. The partitions selected from the input table.	Partition name	No partition is specified by default.
kvDelimiter	Optional. The delimiter used to separate keys and values.	Symbol	The default delimiter is a semicolon (;).
itemDelimiter	Optional. The delimiter used to separate key-value pairs.	Symbol	The default delimiter is a comma (,).
top1200	Optional. This parameter specifies whether to output the first 1,200 columns.	true and false	Default value: true. If the value is false, an error is returned when the number of columns reaches the upper limit.

Parameter	Description	Valid values	Default value
lifecycle	Optional. The lifecycle of the output table.	An integer greater than or equal to -1.	Default value: -1. This value indicates that no lifecycle is set.
coreNum	Optional. The number of cores.	An integer greater than 0.	Default value: -1. This value indicates that the number of instances is determined by the amount of input data.
memSizePerCore	Optional. The memory size of each core.	(100, 65536)	Default value: -1. This value indicates that the memory size is determined by the amount of input data.

Examples

SQL statement to generate data:

```
drop table if exists test;
  create table test as
   select
    *
   from
   (
    select '1:1,2:2,3:-3.3' as kv from dual
    union all
    select '1:10,2:20,3:-33.3' as kv from dual
   ) tmp;
```

PAI command

```
PAI -name KVToTable
  -project algo_public
  -DinputTableName=test
  -DoutputTableName=test_out
  -DoutputKeyMapTableName=test_keymap_out
  -DkvColName=kv;
```

Output

The output table is shown as follows.

```
+------------+------------+------------+
| kv_1     | kv_2     | kv_3     |
+------------+------------+------------+
| 1.0      | 2.0      | -3.3     |
| 10.0     | 20.0     | -33.3    |
+------------+------------+------------+
```

The output mapping table is shown as follows.

```
+------------+------------+------------+
| col_name   | col_index  | col_type   |
+------------+------------+------------+
| kv_1     | 1        | double    |
| kv_2     | 2        | double    |
| kv_3     | 3        | double    |
+------------+------------+------------+
```

Input and output restrictions

Converted columns include appended columns and columns converted from KV pairs. The KV columns are output before the appended columns. MaxCompute supports a maximum of 1,200 columns. When the number of columns exceeds the maximum value, and top1200 is set to true, only the first 1,200 columns are output. If top1200 is set to false, an error is returned. The number of input data entries cannot exceed 100 million.

Restrictions and guidelines

- If a key_map table is input, columns are converted from the keys that exist in both the key_map and key-value tables.
- The converted column type can only be numeric.
- If a key_map table is input, the data type of the converted key column is the same as that of the key_map table. If no key_map table is input, the data type of the converted key column is double.
- If a key_map table is not input, the name of the converted key column is in the format of 'kv column name+""+key'. An error is returned if the key contains any of the following characters: %&()*+-. /;<>=?
- If an appended column is specified and the name of the appended column is the same as that of the converted key column, an error is returned indicating a column name conflict.
- If a row contains multiple keys, the values are added.
- A column name can contain up to 128 characters. If more than 128 characters are entered, only the first 128 characters are kept.

4.4.3.3.7. Table to KV

This component is used to convert data tables to KV tables. Null values in the table to be converted are not displayed in the KV table. You can specify columns to be retained in the new table. These columns will remain unchanged.

PAI command

```
PAI -name TableToKV
    -project algo_public
    -DinputTableName=maple_tabletokv_basic_input
    -DoutputTableName=maple_tabletokv_basic_output
    -DselectedColNames=col0,col1,col2
    -DappendColNames=rowid;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
selectedColNames	Optional. The names of selected columns from the input table.	The column type must be bigint or double.	The whole table is selected by default.
appendColNames	Optional. The names of columns to remain unchanged. These columns are written in the output table without any changes.	Multiple columns can be selected.	-
outputTableName	Required. The name of the output KV table.	Table name	-

Parameter	Description	Valid values	Default value
kvDelimiter	Optional. The delimiter used to separate keys and values.	Symbol	The default delimiter is a colon (:).
itemDelimiter	Optional. The delimiter used to separate key-value pairs.	Symbol	The default delimiter is a comma (,).
convertColToIndexId	Optional. This parameter specifies whether to convert columns into IDs.	0 and 1	0
inputKeyMapTableName	Optional. The name of the input index table. This parameter takes effect only in the case of `convertColToIndexId=1`. If this parameter is not specified, IDs are automatically generated.	Table name	No input index table is set by default.
outputKeyMapTableName	The name of the output index table. This parameter is required only in the case of `convertColToIndexId=1`.	Table name	The default value is determined by convertColToIndexId.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

Example 1

Data generation

rowid	kv
0	col0:1,col1:1.1,col2:2
1	col0:0,col1:1.2,col2:3
2	col0:1,col1:2.3
3	col0:1,col1:0.0,col2:4

PAI command

```
PAI -name TableToKV
    -project algo_public
    -DinputTableName=maple_tabletokv_basic_input
    -DoutputTableName=maple_tabletokv_basic_output
    -DselectedColNames=col0,col1,col2
    -DappendColNames=rowid;
```

Output

The output table is shown as follows.

maple_tabletokv_basic_output

rowid:bigint	kv:string
0	1:1.1,2:2
1	1:1.2,2:3
2	1:2.3
3	1:0.0,2:4

Example 2

PAI command

```
PAI -name TableToKV
    -project projectxlib4 -DinputTableName=maple_tabletokv_basic_input
    -DoutputTableName=maple_tabletokv_basic_output
    -DselectedColNames=col0,col1,col2 -DappendColNames=rowid
    -DconvertColToIndexId=1
    -DinputKeyMapTableName=maple_test_tabletokv_basic_map_input
    -DoutputKeyMapTableName=maple_test_tabletokv_basic_map_output;
```

Output

The output table is shown as follows.

maple_test_tabletokv_basic_map_output

col_name:string	col_index:string	col_datatype:string
col1	1	bigint
col2	2	double

Restrictions and guidelines

- If a key_map table is input, columns are converted from the keys that exist in both the key_map and key-value tables.
- If a key_map table is input and its type is different from the input table, the output key_map table uses the type specified by the user.
- The type of the columns that need to be converted into KV pairs in the input table must be bigint or double.

4.4.4. Feature engineering

4.4.4.1. Feature transformation

4.4.4.1.1. PCA

You can use principal component analysis (PCA) to reduce dimensionality.

- For more information about the PCA algorithm, see Wikipedia.
- This component supports the dense data format.

PAI command

```
PAI -name PrinCompAnalysis
-project algo_public
-DinputTableName=bank_data
-DeigOutputTableName=pai_temp_2032_17900_2
-DprincompOutputTableName=pai_temp_2032_17900_1
-DselectedColNames=pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employ
ed
-DtransType=Simple
-DcalcuType=CORR
-DcontriRate=0.9;
```

Algorithm parameters

Parameters

Parameter	Description	Default value
inputTableName	Required. The name of the input table for PCA.	-
eigOutputTableName	Required. The name of the output table that contains eigenvectors and eigenvalues.	-
princompOutputTableName	Required. The name of the output table after PCA dimensionality reduction.	-
selectedColNames	Required. The names of feature columns that are involved in the PCA procedure.	-
transType	Optional. The method used to transform the original table to the principal component table. Valid values: Simple, Sub-Mean, and Normalization.	Simple
calcuType	Optional. The eigendecomposition mode of the original table. Valid values: CORR, COVAR_SAMP, and COVAR_POP.	CORR
contriRate	Optional. The ratio of information to be retained after dimensionality reduction.	0.9
remainColumns	Optional. The columns retained from the original table after dimensionality reduction.	-

Sample PCA output

Table after dimensionality reduction,shows a sample table after dimensionality reduction.

Eigenvalues and eigenvectors,shows the eigenvalues and eigenvectors.

4.4.4.2. Feature importance evaluation

4.4.4.2.1. Linear model feature importance

You can evaluate the quality of a linear algorithm model based on the predicted and actual output results such as the indicators and residual histogram. Indicators include SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count, yMean, and predictMean.

PAI command

```
pai -name regression_evaluation
-project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DindexOutputTableName=index_output_table
-DresidualOutputTableName=residual_output_table
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training.	-	All partitions in the input table are selected by default.
yColName	Required. The name of the expected dependent variable column in the input table. It must be a numerical value.	-	-
predictionColName	Required. The name of the predicted dependent variable column. It must be a numerical value.	-	-
indexOutputTableName	Required. The name of the regression indicator output table.	-	-
residualOutputTableName	Required. The name of the residual histogram output table.	-	-
intervalNum	Optional. The number of intervals to divide the histogram over.	-	100
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	-	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	-	Automatically calculated.

Output

The output table is in JSON format. Field description describes the JSON fields.

Field description

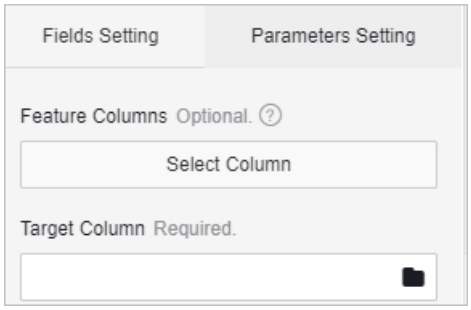
Field	Description

Field	Description
SST	Total sum of squares.
SSE	Sum of squared errors.
SSR	Sum of squares due to regression.
R2	Coefficient of determination.
R	Coefficient of multiple correlation.
MSE	Mean squared error.
RMSE	Root-mean-square error.
MAE	Mean absolute error.
MAD	Mean absolute difference.
MAPE	Mean absolute percentage error.
count	Number of rows.
yMean	Mean of expected dependent variables.
predictionMean	Mean of prediction results.

4.4.4.2.2. Random forest feature importance

You can calculate the importance of features in a random forest model.

Column settings



PAI command

```
pai -name feature_importance
-project algo_public
-DinputTableName=input
-DoutputTableName=output
-Dlabel=label
-DmodelName=model
```

Algorithm parameters

Parameters

Parameter	Description	Default value
inputTableName	Required. The name of the input table.	-
outputTableName	Required. The name of the output table.	-
labelColName	Required. The name of the label column.	-
modelName	Required. The name of the input model.	-
featureColNames	Optional. The names of feature columns selected from the input table.	All columns except the label column are selected by default.
inputTablePartitions	Optional. The partitions selected from the input table.	The whole table is selected by default.
lifecycle	Optional. The lifecycle of the output table.	No lifecycle is set by default.
coreNum	Optional. The number of cores.	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	Automatically calculated.

4.4.5. Statistical analysis

4.4.5.1. Data pivoting

This component allows you to view the distributions of feature values, feature columns, and label columns. Data can be analyzed more efficiently when you know its features. This component supports dense and sparse formats.

PAI command

```
PAI -name fe_meta_runner -project algo_public
-DinputTable="pai_dense_10_10"
-DoutputTable="pai_temp_2263_20384_1"
-DmapTable="pai_temp_2263_20384_2"
-DselectedCols="pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_employed,age,campaign,poutcome"
-DlabelCol="y"
-DcategoryCols="previous"
-Dlifecycle="28"-DmaxBins="5" ;
```

Algorithm parameters

Parameter	Description	Required	Default value
inputTable	The name of the input table.	Yes	N/A
inputTablePartitions	The partitions selected from the input table.	No	N/A
outputTable	The name of the output table.	Yes	N/A
mapTable	The output mapping table. The Data Pivoting component maps String and Int type data for machine learning to use for training.	Yes	N/A
selectedCols	The columns selected from the input table.	Yes	N/A
categoryCols	The columns specified to process Int or Double type columns as enumeration features.	No	null
maxBins	The maximum number of intervals for equal-distance division of continuous features.	No	100
isSparse	Indicates whether the features are in sparse format.	No	false

Parameter	Description	Required	Default value
itemSpliter	The delimiter used to separate sparse feature items.	No	","
kvSpliter	The delimiter used to separate keys and values.	No	":"
lifecycle	The lifecycle of the output table. Unit: days.	No	28

4.4.5.2. Whole table statistics

This component analyzes a table or selected columns of a table.

Parameter settings

In the Input Columns box, select the columns of the table to be analyzed. By default, all columns are selected. You can enter filtering conditions for the selected columns in the condition text box. Operators available include the equal (=), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=) signs, as well as like and rlike.

PAI command

```
PAI -name SimpleSummary
-project algo_public
-DsummaryColNames="euribor3m,pdays"
-DoutputTableNames="pai_temp_667_6017_1"
-DinputTableName="bank_data"
-Dfilter="age>40";
```

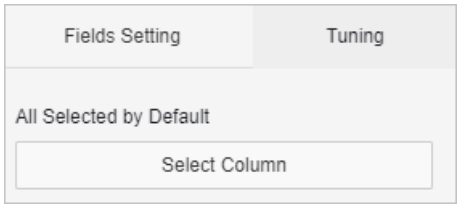
Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
summaryColNames	The columns that require analysis. Separate the columns with commas(,).
outputTableNames	The names of the output tables generated after the system performs the whole table statistics operation.
inputTableName	The name of the input table.
filter	The filtering conditions. Operators available include the equal (=), not equal (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=) signs, as well as like and rlike.

4.4.5.3. Correlation coefficient matrix

The correlation coefficient is a measure of the correlation between columns in a matrix. The valid range of values for this parameter is [-1, 1]. The count equals the number of non-zero elements in two successive columns.

Column settings



PAI command

```
PAI -name corrcoef
-project algo_public
-DinputTableName=maple_test_corrcoef_basic12x10_input
-DoutputTableName=maple_test_corrcoef_basic12x10_output
-DcoreNum=1
-DmemSizePerCore=110
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartition	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions are selected by default.
outputTableName	Required. A list of output table names.	Table name	-
selectedColNames	Optional. The names of columns selected from the input table.	Column name	All columns are selected by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each node. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

Examples

Data generation describes the data generation result.

Data generation

col0:double	col1:bigint	col2:double	col3:bigint	col4:double	col5:bigint	col6:double	col7:bigint	col8:double	col9:double
19	95	33	52	115	43	32	98	76	40
114	26	101	69	56	59	116	23	109	105
103	89	7	9	65	118	73	50	55	81
79	20	63	71	5	24	77	31	21	75
87	16	66	47	25	14	42	99	108	57
11	104	38	37	106	51	3	91	80	97
84	30	70	46	8	6	94	22	45	48
35	17	107	64	10	78	53	34	90	96
13	61	39	1	29	117	112	2	82	28
62	4	102	88	100	36	67	54	12	85
49	27	44	93	68	110	60	72	86	58
92	119	0	113	41	15	74	83	18	111

PAI command

```
PAI -name corrcoef
-project algo_public
-DinputTableName=maple_test_corrcoef_basic12x10_input
-DoutputTableName=maple_test_corrcoef_basic12x10_output
-DcoreNum=1
-DmemSizePerCore=110
```

Output description

Output table

columnsnames	col0	col1	col2
col0	1	-0.2115657251820724	0.0598306259706561
col1	-0.2115657251820724	1	-0.8444477377898585
col2	0.0598306259706561	-0.8444477377898585	1
col3	0.2599903570684693	-0.17507636221594533	0.18518346647293102

columnsnames	col0	col1	col2
col4	-0.3483249188225586	0.40943384150571377	-0.20934839228057014
col5	-0.28716254396809926	0.09135976026101403	-0.1896417512389659
col6	0.47880162127435116	-0.3018506374626574	0.1799377498863213
col7	-0.13646519484213326	0.40733726912808044	-0.3858885676469948
col8	-0.19500158764680092	-0.11827739124590071	0.20254569203773892
col9	0.3897390240949085	0.12433851389455183	0.13476160753756655

4.4.5.4. Covariance

In probability theory and statistics, covariance is a measure of the joint variability of two random variables. Variance is a special case of covariance where the two measured variables are the same. If the expected values are E(X) = μ and E(Y) = v, the covariance between real-number random variables X and Y is cov(X, Y) = E((X - μ) (Y - v)).

PAI command

```
PAI -name cov
    -project algo_public
    -DinputTableName=maple_test_cov_basic12x10_input
    -DoutputTableName=maple_test_cov_basic12x10_output
    -DcoreNum=6
    -DmemSizePerCore=110;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
outputTableName	Required. A list of output table names.	Table name	-
selectedColNames	Optional. The names of columns selected from the input table.	Column name	All columns are selected by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each node. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

4.4.5.5. Empirical probability density chart

An empirical distribution is an estimated non-parametric distribution of probability in scenarios where accurate parametric distributions cannot be made.

The algorithm uses kernel distribution to estimate the probability density of sample data. Similar to a histogram, the algorithm generates functions to describe the distribution of sample data. However, kernel distribution is different in that it overlays the contributions of all parts to generate a smooth and continuous distribution curve, while a histogram only generates discrete descriptions. When kernel distribution is used, the probability density of non-sample data points is not 0, but an overlay of weighted probability density of all sampling points in a certain kernel distribution. In this document, the kernel distribution used is Gaussian distribution.

- For more information about kernel distribution, see Wikipedia.
- For more information about empirical distribution, see Wikipedia.

PAI command

```
PAI -name empirical_pdf
-project algo_public
-DinputTableName="test_data"
-DoutputTableName="test_epdf_out"
-DfeatureColNames="col0,col1,col2"
-DinputTablePartitions="ds='20160101'"
-Dlifecycle=1
-DintervalNum=100
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
outputTableName	Required. The name of the output table.	Table name	-
featureColNames	Required. The names of input columns.	Multiple columns of the double or bigint type can be selected.	-
labelColName	Optional. The name of the input label column.The feature column is stratified based on the label values in the label column.	Only one column of the bigint or string type can be selected. The number of label values cannot exceed 100.	No input label column is set by default.
inputTablePartitions	Optional. The partitions selected from the input table.	Partition name	All partitions are selected by default.
intervalNum	The number of calculation intervals. The larger the number, the higher the accuracy.	[1, 1E14)	Default value: -1. This value indicates that the number of intervals is determined based on the range of data values for each column.
lifecycle	The lifecycle of the output table.	A positive integer	Default value: -1. This value indicates that no lifecycle is set.

Parameter	Description	Valid values	Default value
coreNum	Optional. The number of cores.	A positive integer	Default value: -1. This value indicates that the number of instances is determined based on the volume of input data.
memSizePerCore	Optional. The memory size of each core.	A positive integer in the range of [1024, 65536]	Default value: -1. This value indicates that the memory size is determined based on the volume of input data.

Examples

SQL statement to generate data:

```
drop table if exists epdf_test;
create table epdf_test as
select
 *
from
(
 select 1.0 as col1 from dual
  union all
 select 2.0 as col1 from dual
  union all
 select 3.0 as col1 from dual
  union all
 select 4.0 as col1 from dual
  union all
 select 5.0 as col1 from dual
) tmp;
```

PAI command

```
PAI -name empirical_pdf
-project algo_public
-DinputTableName=epdf_test
-DoutputTableName=epdf_test_out
-DfeatureColNames=col1;
```

Input description

You can select multiple columns to be calculated. You can select a label column and stratify the columns by label. For example, if the label column contains labels 0 and 1, the columns that need to be calculated are stratified into two groups. One group only contains columns with label 0 and the other group only contains columns with label 1. The probability density for each group is then calculated. If no label column is selected, all feature columns are calculated.

Output description

This component outputs a diagram and a result table. The columns in the result table are as follows. If no label column is selected, NULL is output in the label column.

Column name	Data type
colName	string
label	string
x	double
pdf	double

Output table:

```
+------------+------------+------------+------------+
| colname    | label      | x          | pdf        |
+------------+------------+------------+------------+
| col1       | NULL       | 1.0        | 0.12775155176809325 |
| col1       | NULL       | 1.0404050505050506 | 0.1304256933829622 |
| col1       | NULL       | 1.0808101010101012 | 0.13306325897429525 |
| col1       | NULL       | 1.1212151515151518 | 0.1356613897616418 |
| col1       | NULL       | 1.1616202020202024 | 0.1382173796574596 |
| col1       | NULL       | 1.202025252525253 | 0.1407286844875733 |
| col1       | NULL       | 1.2424303030303037 | 0.14319293014274642 |
| col1       | NULL       | 1.2828353535353543 | 0.14560791960033242 |
| col1       | NULL       | 1.3232404040404049 | 0.14797163876379316 |
| col1       | NULL       | 1.3636454545454555 | 0.1502822610772349 |
| col1       | NULL       | 1.404050505050506 | 0.1525381508819247 |
| col1       | NULL       | 1.4444555555555567 | 0.1547378654919243 |
| col1       | NULL       | 1.4848606060606073 | 0.1568801559764068 |
| col1       | NULL       | 1.525265656565658 | 0.15896396664681753 |
| col1       | NULL       | 1.5656707070707085 | 0.16098843325768245 |
| col1       | NULL       | 1.6060757575757592 | 0.1629528799404685 |
| col1       | NULL       | 1.6464808080808098 | 0.16485681490034038 |
| col1       | NULL       | 1.6868858585858604 | 0.16669992491584543 |
| col1       | NULL       | 1.727290909090911 | 0.16848206869138338 |
| col1       | NULL       | 1.7676959595959616 | 0.17020326912168932 |
```

col1	NULL	1.8081010101010122	0.17186370453638117
col1	NULL	1.8485060606060628	0.17346369900080946
col1	NULL	1.8889111111111134	0.17500371175692428
col1	NULL	1.929316161616164	0.17648432589456017
col1	NULL	1.9697212121212146	0.17790623634938396
col1	NULL	2.0101262626262653	0.1792702373286898
col1	NULL	2.050531313131316	0.18057720927022053
col1	NULL	2.0909363636363665	0.18182810544221673
col1	NULL	2.131341414141417	0.18302393829491406
col1	NULL	2.1717464646464677	0.18416576567472337
col1	NULL	2.2121515151515183	0.1852546770123305
col1	NULL	2.252556565656569	0.18629177959496213
col1	NULL	2.2929616161616195	0.18727818503109434
col1	NULL	2.33336666666667	0.18821499601297229
col1	NULL	2.3737717171717208	0.18910329347850022
col1	NULL	2.4141767676767714	0.18994412426940221
col1	NULL	2.454581818181822	0.19073848937711185
col1	NULL	2.4949868686868726	0.19148733286168018
col1	NULL	2.535391919191923	0.1921915315221827
col1	NULL	2.575796969696974	0.19285188538972659
col1	NULL	2.6162020202020244	0.19346910910630113
col1	NULL	2.656607070707075	0.19404382424446043
col1	NULL	2.6970121212121256	0.1945765526142701
col1	NULL	2.7374171717171762	0.19506771059517916
col1	NULL	2.777822222222227	0.19551760452158667
col1	NULL	2.8182272727272775	0.19592642714194602
col1	NULL	2.858632323232328	0.1962942551623821
col1	NULL	2.8990373737373787	0.1966210478770638
col1	NULL	2.9394424242424293	0.1969066468790639
col1	NULL	2.97984747474748	0.19715077683721793
col1	NULL	3.0202525252525305	0.19735304731663747
col1	NULL	3.060657575757581	0.19751295561309964
col1	NULL	3.1010626262626317	0.19762989056457925
col1	NULL	3.1414676767676823	0.19770313729675995
col1	NULL	3.181872727272733	0.19773188285349683
col1	NULL	3.2222777777777836	0.19771522265793107
col1	NULL	3.262682828282834	0.19765216774530828
col1	NULL	3.303087878787885	0.19754165270453194
col1	NULL	3.3434929292929354	0.19738254426210697
col1	NULL	3.383897979797986	0.19717365043938664
col1	NULL	3.4243030303030366	0.19691373021193162

```
| col1     | NULL     | 3.4647080808080872 | 0.1966015035982942 |
| col1     | NULL     | 3.505113131313138 | 0.19623566210464843 |
| col1     | NULL     | 3.5455181818181885 | 0.19581487945135703 |
| col1     | NULL     | 3.585923232323239 | 0.19533782250778076 |
| col1     | NULL     | 3.6263282828282897 | 0.1948031623623475 |
| col1     | NULL     | 3.6667333333333403 | 0.1942095854560816 |
| col1     | NULL     | 3.707138383838391 | 0.19355580470939734 |
| col1     | NULL     | 3.7475434343434415 | 0.19284057057394655 |
| col1     | NULL     | 3.787948484848492 | 0.19206268194364004 |
| col1     | NULL     | 3.8283535353535427 | 0.19122099686158253 |
| col1     | NULL     | 3.8687585858585933 | 0.19031444296253852 |
| col1     | NULL     | 3.909163636363644 | 0.1893420275936375 |
| col1     | NULL     | 3.9495686868686946 | 0.18830284755928747 |
| col1     | NULL     | 3.989973737373745 | 0.1871960984396676 |
| col1     | NULL     | 4.030378787878796 | 0.18602108343567092 |
| col1     | NULL     | 4.070783838383846 | 0.18477722169674377 |
| col1     | NULL     | 4.111188888888897 | 0.1834640560916829 |
| col1     | NULL     | 4.151593939393948 | 0.1820812603860928 |
| col1     | NULL     | 4.191998989898998 | 0.18062864579383914 |
| col1     | NULL     | 4.232404040404049 | 0.179106166873458 |
| col1     | NULL     | 4.272809090909099 | 0.17751392674406796 |
| col1     | NULL     | 4.31321414141415 | 0.17585218159888508 |
| col1     | NULL     | 4.353619191919201 | 0.17412134449794325 |
| col1     | NULL     | 4.394024242424251 | 0.1723219884250765 |
| col1     | NULL     | 4.434429292929302 | 0.17045484859762067 |
| col1     | NULL     | 4.4748343434343525 | 0.16852082402064342 |
| col1     | NULL     | 4.515239393939403 | 0.1665209782808102 |
| col1     | NULL     | 4.555644444444454 | 0.16445653957824907 |
| col1     | NULL     | 4.596049494949504 | 0.16232889999798905 |
| col1     | NULL     | 4.636454545454555 | 0.16013961402571825 |
| col1     | NULL     | 4.6768595959596055 | 0.1578903963157465 |
| col1     | NULL     | 4.717264646464656 | 0.15558311872216193 |
| col1     | NULL     | 4.757669696969707 | 0.1532198066072439 |
| col1     | NULL     | 4.798074747474757 | 0.1508026344442397 |
| col1     | NULL     | 4.838479797979808 | 0.14833392073462115 |
| col1     | NULL     | 4.878884848484859 | 0.14581612226291346 |
| col1     | NULL     | 4.919289898989909 | 0.1432518277151203 |
| col1     | NULL     | 4.95969494949496 | 0.1406437506896507 |
| col1     | NULL     | 5.00010000000001 | 0.13799472213247665 |
+----------+----------+----------+----------+
```

Input and output restrictions

The maximum number of label columns that can be specified is 100.

4.4.5.6. Chi-square goodness of fit test

This component is used to determine the differences between the observed frequencies and the expected frequencies for each classification of a single multiclass classification nominal variable. The null hypothesis assumes that the observed frequencies and the expected frequencies are consistent.

PAI command

```
PAI -name chisq_test
    -project algo_public
    -DinputTableName=pai_chisq_test_input
    -DcolName=f0
    -DprobConfig=0:0.3,1:0.7
    -DoutputTableName=pai_chisq_test_output0
    -DoutputDetailTableName=pai_chisq_test_output0_detail
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
colName	Required. The name of the column that requires a chi-square test.	Column name	-
outputTableName	Required. The name of the output table.	Table name that has not been used	-
outputDetailTableName	Required. The name of the output detail table.	Table name that has not been used	-
inputTablePartitions	Optional. The partitions selected from the input table.	Partition list	All partitions are selected by default.
probConfig	Optional. The class probability configuration.	The configuration is stored in a key-value pair format: `class:probability`. The sum of all probabilities is 1.	All classes have the same probability by default.

Examples

Testing data

```
create table pai_chisq_test_input as
select * from
(
 select '1' as f0,'2' as f1 from dual
 union all
 select '1' as f0,'3' as f1 from dual
 union all
 select '1' as f0,'4' as f1 from dual
 union all
 select '0' as f0,'3' as f1 from dual
 union all
 select '0' as f0,'4' as f1 from dual
)tmp;
```

PAI command

```
PAI -name chisq_test
   -project algo_public
   -DinputTableName=pai_chisq_test_input
   -DcolName=f0
   -DprobConfig=0:0.3,1:0.7
   -DoutputTableName=pai_chisq_test_output0
   -DoutputDetailTableName=pai_chisq_test_output0_detail
```

Output description

Output table outputTableName is a JSON array containing only one row and one column.

```
{
   "Chi-Square": {
      "comment": "Pearsons chi-square test",
      "df": 1,
      "p-value": 0.75,
      "value": 0.2380952380952381
   }
}
```

Output table outputDetailTableName includes the following columns: data source class (**f0** or **f1**), observed frequency (**observed**), expected frequency (**expected**), and standard residuals (`residuals = (observed - expected)/sqrt (expected)`).

f0	f1	observed	expected	residuals
0	2	0.0	0.4	-0.6324555320336759
0	3	1.0	0.8	0.22360679774997894
0	4	1.0	0.8	0.22360679774997894
1	2	1.0	0.6000000000000001	0.5163977794943221
1	3	1.0	1.2000000000000002	-0.1825741858350555
1	4	1.0	1.2000000000000002	-0.1825741858350555

4.4.5.7. Chi-square test of independence

This component verifies whether two factors (each having two or more classes) are mutually independent. The null hypothesis is that two factors are independent of each other.

PAI command

```
PAI -name chisq_test
    -project algo_public
    -DinputTableName=pai_chisq_test_input
    -DxColName=f0
    -DyColName=f1
    -DoutputTableName=pai_chisq_test_output2
    -DoutputDetailTableName=pai_chisq_test_output2_detail
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
xColName	Required. The name of the column that requires a chi-square test.	Column name	-
yColName	Required. The name of the column that require a chi-square test.	Column name	-
outputTableName	Required. The name of the output table.	Table name that has not been used	-
outputDetailTableName	Required. The name of the output detail table.	Table name that has not been used	-

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table.	Partition list	All partitions are selected by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Examples

- Testing data

```
create table pai_chisq_test_input as
select * from
(
select '1' as f0,'2' as f1 from dual
union all
select '1' as f0,'3' as f1 from dual
union all
select '1' as f0,'4' as f1 from dual
union all
select '0' as f0,'3' as f1 from dual
union all
select '0' as f0,'4' as f1 from dual
)tmp;
```

- PAI command

```
PAI -name chisq_test
  -project algo_public
  -DinputTableName=pai_chisq_test_input
  -DxColName=f0
  -DyColName=f1
  -DoutputTableName=pai_chisq_test_output2
  -DoutputDetailTableName=pai_chisq_test_output2_detail
```

- Output description

  Output table outputTableName is a JSON array containing only one row and one column.

```
{
  "Chi-Square": {
    "comment": "Pearsons chi-square test",
    "df": 2,
    "p-value": 0.75,
    "value": 0.8333333333333334
  }
}
```


Output table outputDetailTableName has the following columns:

Column name	Description
xColName	Class
yColName	Class
observed	Observed frequency
expected	Expected frequency
residuals	Residuals = (observed - expected)/sqrt (expected)

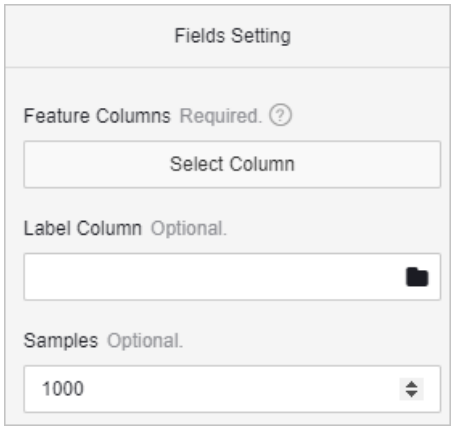
Data:

f0	f1	observed	expected	residuals
0	2	0.0	0.4	-0.6324555320336759
0	3	1.0	0.8	0.22360679774997894
0	4	1.0	0.8	0.22360679774997894
1	2	1.0	0.6000000000000001	0.5163977794943221
1	3	1.0	1.2000000000000002	-0.1825741858350555
1	4	1.0	1.2000000000000002	-0.1825741858350555

4.4.5.8. Scatter plot

In regression analysis, this component outputs a scatter plot that shows the distribution of data points in a Cartesian coordinate system.

Column settings



PAI command

```
PAI -name scatter_diagram
-project algo_public
-DselectedCols=emp_var_rate,cons_price_rate,cons_conf_idx,euribor3m
-DsampleSize=1000
-DlabelCol=y
-DmapTable=pai_temp_2447_22859_2
-DinputTable=scatter_diagram
-DoutputTable=pai_temp_2447_22859_1
```

Parameters

Parameter	Description	Default value
inputTable	Required. The name of the input table.	-
inputTablePartitions	Optional. The partitions selected from the input table.	-
outputTable	Required. The name of the output table that stores the samples.	-
mapTable	Required. The name of the output table that stores the maximum value, minimum value, and enumeration values of each feature.	-
selectedCols	Required. The columns selected from the input table from which to draw a scatter plot. A maximum of five features can be selected.	-
labelCol	Optional. An Int or String column to serve as the enumeration label column.	No enumeration label column is set by default.

Parameter	Description	Default value
sampleSize	Optional. The number of samples to collect from the input data.	1000
lifecycle	Optional. The lifecycle of the output table measured in days.	28

Examples

Input data

```
create table scatter_diagram as select emp_var_rate,cons_price_rate, cons_conf_idx,euribor3m,y from pai_bank_data limit 10
```

Parameters

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
1.4	93.918	-42.7	4.962	0
-0.1	93.2	-42.0	4.021	0
-1.7	94.055	-39.8	0.729	1
-1.8	93.075	-47.1	1.405	0
-2.9	92.201	-31.4	0.869	1
1.4	93.918	-42.7	4.961	0
-1.8	92.893	-46.2	1.327	0
-1.8	92.893	-46.2	1.313	0
-2.9	92.963	-40.8	1.266	1
-1.8	93.075	-47.1	1.41	0
1.1	93.994	-36.4	4.864	0
1.4	93.444	-36.1	4.964	0
1.4	93.444	-36.1	4.965	1
-1.8	92.893	-46.2	1.291	0
1.4	94.465	-41.8	4.96	0
1.4	93.918	-42.7	4.962	0
-1.8	93.075	-47.1	1.365	1

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
-0.1	93.798	-40.4	4.86	1
1.1	93.994	-36.4	4.86	0
1.4	93.918	-42.7	4.96	0
-1.8	93.075	-47.1	1.405	0
1.4	94.465	-41.8	4.967	0
1.4	93.918	-42.7	4.963	0
1.4	93.918	-42.7	4.968	0
1.4	93.918	-42.7	4.962	0
-1.8	92.893	-46.2	1.344	0
-3.4	92.431	-26.9	0.754	0
-1.8	93.075	-47.1	1.365	0
-1.8	92.893	-46.2	1.313	0
1.4	93.918	-42.7	4.961	0
1.4	94.465	-41.8	4.961	0
-1.8	92.893	-46.2	1.327	0
-1.8	92.893	-46.2	1.299	0
-2.9	92.963	-40.8	1.268	1
1.4	93.918	-42.7	4.963	0
-1.8	92.893	-46.2	1.334	0
1.4	93.918	-42.7	4.96	0
-1.8	93.075	-47.1	1.405	0
1.4	94.465	-41.8	4.96	0
1.4	93.444	-36.1	4.962	0
1.1	93.994	-36.4	4.86	0
1.1	93.994	-36.4	4.857	0
1.4	93.918	-42.7	4.961	0

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
-3.4	92.649	-30.1	0.715	1
1.4	93.444	-36.1	4.966	0
-0.1	93.2	-42.0	4.076	0
1.4	93.444	-36.1	4.965	0
-1.8	92.893	-46.2	1.354	0
1.4	93.444	-36.1	4.967	0
1.4	94.465	-41.8	4.959	0
-1.8	92.893	-46.2	1.354	0
1.4	94.465	-41.8	4.958	0
-1.8	92.893	-46.2	1.354	0
1.4	94.465	-41.8	4.864	0
1.1	93.994	-36.4	4.859	0
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.27	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.859	0
1.4	94.465	-41.8	4.959	0
1.1	93.994	-36.4	4.856	0
-1.8	93.075	-47.1	1.405	0
-1.8	92.843	-50.0	1.811	1
-0.1	93.2	-42.0	4.021	0
-2.9	92.469	-33.6	1.029	0
1.4	93.918	-42.7	4.962	0
-1.8	93.075	-47.1	1.365	0
1.1	93.994	-36.4	4.857	0
-1.8	92.893	-46.2	1.259	0

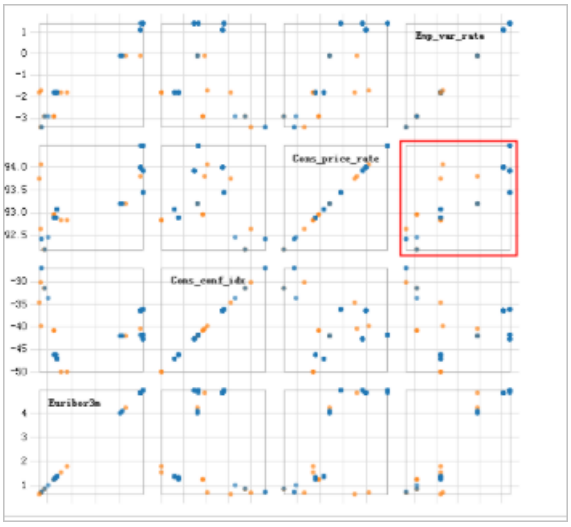
emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
1.1	93.994	-36.4	4.857	0
1.4	94.465	-41.8	4.866	0
-2.9	92.201	-31.4	0.883	0
-0.1	93.2	-42.0	4.076	0
1.1	93.994	-36.4	4.857	0
1.4	93.918	-42.7	4.96	0
1.4	93.444	-36.1	4.962	0
1.1	93.994	-36.4	4.858	0
1.1	93.994	-36.4	4.857	0
1.1	93.994	-36.4	4.856	0
1.4	93.918	-42.7	4.968	0
1.4	93.444	-36.1	4.966	0
1.4	94.465	-41.8	4.962	0
1.4	93.444	-36.1	4.963	0
-1.8	92.843	-50.0	1.56	1
1.4	93.918	-42.7	4.96	0
1.4	93.444	-36.1	4.963	0
-3.4	92.431	-26.9	0.74	0
1.1	93.994	-36.4	4.856	0
1.4	93.918	-42.7	4.962	0
1.1	93.994	-36.4	4.856	0
-0.1	93.2	-42.0	4.245	1
1.1	93.994	-36.4	4.857	0
-1.8	93.075	-47.1	1.405	0
-1.8	92.893	-46.2	1.327	0
-0.1	93.2	-42.0	4.12	0

emp_var_rate	cons_price_rate	cons_conf_idx	euribor3m	y
1.4	94.465	-41.8	4.958	0
-1.8	93.749	-34.6	0.659	1
1.1	93.994	-36.4	4.858	0
1.1	93.994	-36.4	4.858	0
1.4	93.444	-36.1	4.963	0

Parameter settings

Scatter plot configuration: select **select emp_var_rate**, **cons_price_rate**, **cons_conf_idx**, and **euribor3m** as the feature columns, and select **y** as the label column.

Output



You can view the distribution of classification tags between every two features in the scatter plot.

4.4.5.9. Two-sample T-test

A two-sample T-test is composed of an independent sample T-test and a paired sample T-test. Two samples independent of each other are called independent samples. An independent sample T-test checks whether two samples are significantly different from each other. The T-test is based on the premise that two samples are independent of each other and come from two normally distributed populations. A paired sample T-test checks whether the mean values from two paired populations are significantly different from each other.

PAI command

```
PAI -name t_test
    -project algo_public
    -DxTableName=pai_t_test_all_type
    -DxColName=col1_double
    -DxTablePartitions=ds=2010/dt=1
    -DyTableName=pai_t_test_all_type
    -DyColName=col1_double
    -DyTablePartitions=ds=2010/dt=1
    -DoutputTableName=pai_t_test_out
    -Dalternative=less
    -Dmu=47
    -DconfidenceLevel=0.95
    -Dpaired=False
    -DvarEqual=True
```

Parameters

Parameter	Description	Valid values	Default value
xTableName	Required. The name of input table x.	Table name	-
xTablePartitions	Optional. The partitions selected from input table x for testing, in the format of `Partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
xColName	Required. The column selected from table x for testing.	Column name. The type must be double or bigint.	-
yTableName	Required. The name of input table y.	Table name	-
yTablePartitions	Optional. The partitions selected from input table y for testing, in the format of `Partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.

Parameter	Description	Valid values	Default value
yColName	Required. The name of the column selected from table y for testing.	Column name. The type must be double or bigint.	-
paired	Optional. A value of True indicates that it is a paired sample T-test. A value of False indicates that it is an independent sample T-test.	True and False	False
alternative	Optional. The alternative hypothesis.	two.sided, less, and greater	two.sided
mu	Optional. The hypothesized mean.	double	0
varEqual	Optional. This parameter indicates whether two population variances are equal.	True and False	False
confidenceLevel	Optional. The confidence level.	0.8, 0.9, 0.95, 0.99, 0. 995, and 0.999	0.95
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each node. Unit: MB.	A positive integer in the range of [1024, 65536].	Automatically calculated.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Custom resources

Note

- For a regular table, we recommend that you do not set coreNum and memSizePerCore, and instead allow the default values to be used automatically.
- If you do not have sufficient compute resources, use the following code to calculate the amount of compute resources needed:

```
def CalcCoreNumAndMem(row, centerCount, kOneCoreDataSize=1024):
  """Calculates the number of nodes and memory size needed for each node.
    Args:
      row: the number of rows in the input table.
      col: the number of columns in the input table.
      kOneCoreDataSize: the amount of data needed to be calculated per node. Unit: MB. The value m
ust be a positive integer. Default value: 1024.
    Return:
      coreNum, memSizePerCore
    Example:
      coreNum, memSizePerCore = CalcCoreNumAndMem(1000,99, 100, kOneCoreDataSize=2048)

  """
  kMBytes = 1024.0 * 1024.0
  #Number of compute nodes
  coreNum = max(1, int(row * 2 * 8 / kMBytes / kOneCoreDataSize))
  #Memory size per node = Data volume
  memSizePerCore = max(1024,  int(kOneCoreDataSize*2))
  return coreNum,  memSizePerCore
```

Examples

- SQL statement to generate data:

```
create table pai_test_input as
select * from
(
 select 1 as f0,2 as f1 from dual
 union all
 select 1 as f0,3 as f1 from dual
 union all
 select 1 as f0,4 as f1 from dual
 union all
 select 0 as f0,3 as f1 from dual
 union all
 select 0 as f0,4 as f1 from dual
)tmp;
```

- PAI command

```
PAI -name t_test
    -project algo_public
    -DxTableName=pai_test_input
    -DxColName=f0
    -DyTableName=pai_test_input
    -DyColName=f1
    -DyTablePartitions=ds=2010/dt=1
    -DoutputTableName=pai_t_test_out
    -Dalternative=less
    -Dmu=47
    -DconfidenceLevel=0.95
    -Dpaired=False
    -DvarEqual=True
```

- Output description

  The output table is a JSON array containing only one row and one column.

```
{
    "AlternativeHypthesis": "difference in means not equals to 0",
    "ConfidenceInterval": "(-2.5465, -0.4535)",
    "ConfidenceLevel": 0.95,
    "alpha": 0.05000000000000004,
    "df": 19,
    "mean of the differences": -1.5,
    "p": 0.008000000000000007,
    "t": -3
}
```

Input and output restrictions

The input and output are not limited.

4.4.5.10. One-sample T-test

A one-sample T-test verifies whether the mean of a normally distributed population differs significantly from a target value. A T-test is performed based on the condition that the sample population is normally distributed.

PAI command

```
PAI -name t_test -project algo_public
    -DxTableName=pai_t_test_all_type
    -DxColName=col1_double
    -DoutputTableName=pai_t_test_out
    -DxTablePartitions=ds=2010/dt=1
    -Dalternative=less
    -Dmu=47
    -DconfidenceLevel=0.95
```

Algorithm parameters

Parameter	Description	Valid values	Default value
xTableName	Required. The name of input table x.	Table name	-
xColName	Required. The column selected from table x for testing.	Column name. The type must be double or bigint.	-
outputTableName	Required. The name of the output table.	Table name that has not been used	-
xTablePartitions	Optional. The partitions selected from input table x.	Partition list	All partitions are selected by default.
alternative	Optional. The alternative hypothesis.	two.sided, less, and greater	two.sided
mu	Optional. The hypothesized mean.	double	0
confidenceLevel	Optional. The confidence level.	0.8, 0.9, 0.95, 0.99, 0.995, and 0.999	0.95

Output description

The output table is a JSON array containing only one row and one column.

```
{
   "AlternativeHypthesis": "mean not equals to 0",
   "ConfidenceInterval": "(44.72234194006504, 46.27765805993496)",
   "ConfidenceLevel": 0.95,
   "alpha": 0.05,
   "df": 99,
   "mean": 45.5,
   "p": 0,
   "stdDeviation": 3.919647479510927,
   "t": 116.081867662439
}
```

4.4.5.11. Lorenz curve

The Lorenz curve is a graph to illustrate the distribution of wealth across a population. The X axis represents the total population arranged from least wealthy to most wealthy, while the Y axis represents the total wealth. If this graph is a straight line, it indicates perfectly equal distribution of wealth. The Gini coefficient is calculated by taking the area between the equal distribution curve and the actual Lorenz curve for a population as a fraction of the total area beneath the equal distribution curve. As the distribution of wealth becomes less equal, the Gini coefficient will increase, whereas a population with equal distribution of wealth will have a Gini coefficient of 0.

To study the distribution of income among a population, American statistician Max Otto Lorenz proposed the famous Lorenz curve in 1905. In 1921, Italian economist Corrado Gini defined the Gini coefficient as a measure of inequality in a population based on the Lorenz curve.

PAI command

```
PAI -name LorenzCurve
   -project algo_public
   -DinputTableName=maple_test_lorenz_basic10_input
   -DcolName=col0
   -DoutputTableName=maple_test_lorenz_basic10_output -DcoreNum=20
   -DmemSizePerCore=110;
```

Parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	Table name	N/A
outputTableName	Required. The name of the output table.	Table name that has not been used	N/A

Parameter	Description	Valid value	Default value
colName	Optional. The column name. Separate multiple columns with commas (,).	Column name	The whole table is selected by default.
N	The number of quantiles.	N/A	100
inputPartitions	Optional. The partitions selected from the input table for training, in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

Examples

Data generation

col0:double
4
7
2
8
6

col0:double
3
9
5
0
1
10

PAI command

```
PAI -name LorenzCurve
    -project algo_public
    -DinputTableName=maple_test_lorenz_basic10_input
    -DcolName=col0
    -DoutputTableName=maple_test_lorenz_basic10_output
    -DcoreNum=20
    -DmemSizePerCore=110;
```

Output

Quantile	col0
0	0
1	0.01818181818181818
2	0.01818181818181818
3	0.01818181818181818
4	0.01818181818181818
5	0.01818181818181818
6	0.01818181818181818
7	0.01818181818181818
8	0.01818181818181818
9	0.01818181818181818
10	0.01818181818181818
11	0.05454545454545454

Quantile	col0
12	0.05454545454545454
13	0.05454545454545454
14	0.05454545454545454
...	...
85	0.8181818181818182
86	0.8181818181818182
87	0.8181818181818182
88	0.8181818181818182
89	0.8181818181818182
90	1
91	1
92	1
93	1
94	1
95	1
96	1
97	1
98	1
99	1
100	1

4.4.5.12. Normality test

This component is used to determine whether observed values are normally distributed.

This component consists of three test methods: Anderson-Darling test (see Wikipedia), Kolmogorov-Smirnov test (see Wikipedia), and Q-Q plot (see Wikipedia). You can use one or more methods as needed.

Algorithm description:

- Original hypothesis H0: The observed values are normally distributed. H1: The observed values are not normally distributed.
- The KS p-value calculation method progressively calculates CDF of KS distribution regardless of the sample size. For more information, see Wikipedia.
- If the sample size is greater than 1000, the Q-Q plot method collects samples to calculate and output plots. This means that the data points in plots do not necessarily cover all samples.

PAI command

```
PAI -name normality_test
    -project algo_public
    -DinputTableName=test
    -DoutputTableName=test_out
    -DselectedColNames=col1,col2
    -Dlifecycle=1;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
outputTableName	Required. The name of the output table.	Table name that has not been used	-
selectedColNames	Optional. The names of selected columns.	Multiple double or bigint type columns can be selected.	-
inputTablePartitions	Optional. The partitions selected from the input table.	Partition name	All partitions are selected by default.
enableQQplot	Optional. This parameter specifies whether to use the Q-Q plot.	true and false	true
enableADtest	Optional. This parameter specifies whether to perform the Anderson-Darling test.	true and false	true
enableKStest	Optional. This parameter specifies whether to perform the Kolmogorov-Smirnov test.	true and false	true

Parameter	Description	Valid values	Default value
lifecycle	Optional. The lifecycle of the output table.	An integer greater than or equal to -1	Default value: -1. This value indicates that no lifecycle is set.
coreNum	Optional. The number of cores.	An integer greater than 0	Default value: -1. This value indicates that the number of instances is determined by the amount of input data.
memSizePerCore	Optional. The memory size of each core.	(100, 65536)	Default value: -1. This value indicates that the memory size is determined by the amount of input data.

Examples

- **SQL statement to generate data:**

```
drop table if exists normality_test_input;
  create table normality_test_input as
  select
   *
  from
  (
   select 1 as x from dual
    union all
   select 2 as x from dual
    union all
   select 3 as x from dual
    union all
   select 4 as x from dual
    union all
   select 5 as x from dual
    union all
   select 6 as x from dual
    union all
   select 7 as x from dual
    union all
   select 8 as x from dual
    union all
   select 9 as x from dual
    union all
   select 10 as x from dual
  ) tmp;
```

- **PAI command**

```
PAI -name normality_test
  -project projectxlib4
  -DinputTableName=normality_test_input
  -DoutputTableName=normality_test_output
  -DselectedColNames=x
  -Dlifecycle=1;
```

- **Input description**

  Input format: select the columns that need to be calculated. The columns must be of the double or bigint type.

- **Output description**

  A diagram and a result table are output. The columns in the result table are as follows. The result table has two partitions:

- `p='test'` shows the result of the AD or KS test. Data is output when *enableADtest* or *enableKStest* is set to true.
- `p='plot'` shows the Q-Q plot data. When *enableQQplot* is set to true, data is output and the columns that meet the `p='test'` condition are reused. In the case of `p='plot'`, the *testvalue* column records the original observed data (x axis of the Q-Q plot), and the *pvalue* column records the expected data that is normally distributed (y axis of the Q-Q plot).

Output table:

```
+------------+------------+------------+------------+------------+
| colname    | testname   | testvalue  | pvalue     | p          |
+------------+------------+------------+------------+------------+
| x          | NULL       | 1.0        | 0.8173291742279805 | plot       |
| x          | NULL       | 2.0        | 2.470864450785345 | plot       |
| x          | NULL       | 3.0        | 3.5156067948020056 | plot       |
| x          | NULL       | 4.0        | 4.3632330349313095 | plot       |
| x          | NULL       | 5.0        | 5.128868067945126 | plot       |
| x          | NULL       | 6.0        | 5.871131932054874 | plot       |
| x          | NULL       | 7.0        | 6.6367669650686905 | plot       |
| x          | NULL       | 8.0        | 7.4843932051979944 | plot       |
| x          | NULL       | 9.0        | 8.529135549214654 | plot       |
| x          | NULL       | 10.0       | 10.182670825772018 | plot       |
| x          | Anderson_Darling_Test | 0.1411092332197832 | 0.9566579606430077 | test       |
| x          | Kolmogorov_Smirnov_Test | 0.09551932503797644 | 0.9999888659426232 | test       |
+------------+------------+------------+------------+------------+
```

Column name	Data type	Definition
colName	string	Column name
testname	string	Test name
testvalue	double	Test value on the x axis of the Q-Q plot
pvalue	double	Test p value on the y axis of the Q-Q plot
p	double	Partition name

4.4.5.13. Percentile

This component calculates the percentile of the values in a column.

Parameter settings

Select the column to be analyzed. Only the double and bigint types are supported.

PAI command

```
PAI -name Percentile
-project algo_public
-DoutputTableName="pai_temp_666_6014_1"
-DcolName="euribor3m"
-DinputTableName="bank_data";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
outputTableName	The name of the output table automatically generated after the system performs the percentile calculation.
colName	The column selected for percentile calculation. Only the numeric type is supported.
inputTableName	The name of the input table.

4.4.5.14. Pearson coefficient

This component calculates the Pearson correlation coefficient of two numeric columns in an input table or a partition, and saves the result to the output table.

Component description

- The component has only two parameters: input column 1 and input column 2. Enter the names of the two columns for which the Pearson correlation coefficient is calculated.
- After you run the component, right-click the component and choose **View Analytics Report** from the shortcut menu.
- The Pearson correlation coefficient is listed in the row.

PAI command

```
pai -name pearson
-project algo_test
-DinputTableName=wpbc
-Dcol1Name=f1
-Dcol2Name=f2
-DoutputTableName=wpbc_pear;
```

Algorithm prameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartitions	The partitions selected from the input table for calculation.	The parameter value must be in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	All partitions in the input table are selected by default.
col1Name	Required. The name of input column 1.	Column name	-
col2Name	Required. The name of input column 2.	Column name	-
outputTableName	Required. The name of the output table.	Table name	-

4.4.5.15. Histogram

This component analyzes data in a column and outputs a histogram.

Parameter settings

- Select the columns to be analyzed. Only the double and bigint types are supported.
- View the analysis report.
- You can adjust the step and move the slider to view the entire histogram.

4.4.6. Machine learning

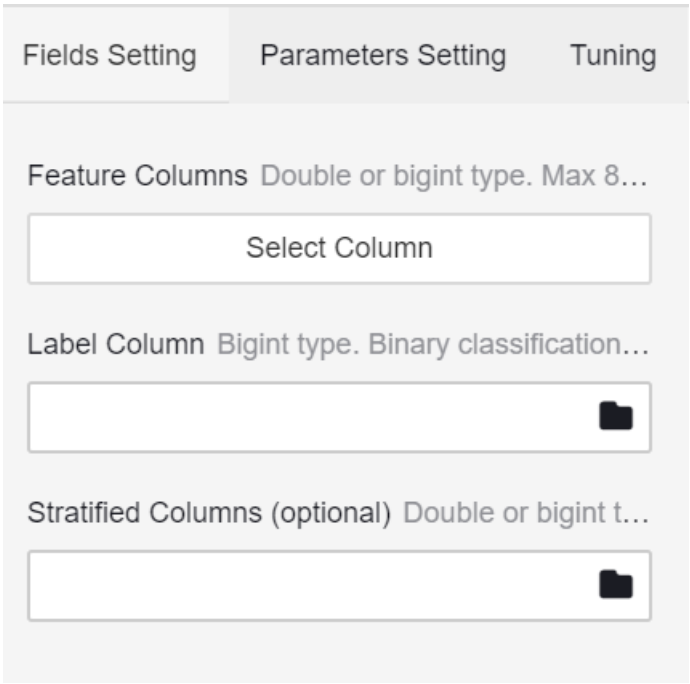
4.4.6.1. Binary classification

4.4.6.1.1. GBDT binary classification

This component is used for binary classification based on GBDT regression and sorting. Values greater than the threshold value are considered positive samples, while values that are less than or equal to the threshold value are considered negative samples.

Procedure

1. Drag and drop the GBDT Binary Classification component onto the canvas for training and set the parameters, as shown in the following figure.



Parameters

Parameter	Description
Feature Columns	The double and bigint types are supported. A maximum of 800 columns can be specified.
Label Column	You can select all columns except the input column. The values must be of the binary type.
Stratification Column	Optional. The whole table is selected by default. The double and bigint types are supported.

2. You can change the data type of the input columns.

   The input columns of GBDT binary classification only support the continuous type and are processed in the same way as the discrete type.

3. Set the parameters.

Parameters

Parameter	Description
Metric Type	The normalized discounted cumulative gain (NDCG) and discounted cumulative gain (DCG).
Trees	Valid values: [1,10000]. Default value: 500.
Learning Rate	Valid values: (0, 1). Default value: 0.05.
Training Sample Fraction	Valid values: (0, 1). Default value: 0.6.
Training Feature Fraction	Valid values: (0, 1). Default value: 0.6.
Maximum Leaves	The value must be an integer in the range of [2, 1000]. Default value: 32.
Testing Data Fraction	Valid values: [0, 1]. Default value: 0.0.
Maximum Tree Depth	The value must be an integer in the range of [1, 11]. Default value: 11.
Minimum Samples per Leaf Node	The value must be an integer in the range of [100, 1000]. Default value: 500.
Random Seed	The value must be an integer in the range of [0, 10]. Default value: 0.
Maximum Splits per Feature	Valid values: [1, 1000]. Default value: 500.

4. View the output. For more information, see the description of the Random forest component.

> Note
>
> - GBDT and GBDT_LR have different default types of loss functions. The default loss function of GBDT is regression loss:mean squared error loss. The default loss function of GBDT_LR is logistic regression loss. The system automatically writes the default loss function for GBDT_LR.
> - For GBDT binary classification, the label column must be of the binary type. String type data is not supported.
> - When connecting the ROC curve component, set the prediction component parameters and select a base value.

PAI command (F/L setup settings are not used)

```
PAI -name GBDT_LR
-project algo_public
-DfeatureSplitValueMaxSize="500"
-DrandSeed="0"
-Dshrinkage="0.5"
-DmaxLeafCount="32"
-DlabelColName="y"
-DinputTableName="bank_data_partition"
-DminLeafSampleCount="500"
-DgroupIDColName="nr_employed"
-DsampleRatio="0.6"
-DmaxDepth="11"
-DmodelName="xlab_m_GBDT_LR_21208"
-DmetricType="2"
-DfeatureRatio="0.6"
-DinputTablePartitions="pt=20150501"
-DtestRatio="0.0"
-DfeatureColNames="age,previous,cons_conf_idx,euribor3m"
-DtreeCount="500";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
featureSplitValueMaxSize	Optional. The maximum number of splits per feature. Valid values: [1, 1000]. Default value: 500.
randSeed	Optional. The number of random seeds. The value must be an integer in the range of [0, 10]. Default value: 0.
shrinkage	Optional. The learning rate. Valid values: (0, 1). Default value: 0.05.
maxLeafCount	Optional. The maximum number of leaves. The value must be an integer in the range of [2, 1000]. Default value: 32.
labelColName	The name of the label column selected from the input table.
inputTableName	The name of the input table for training.

Parameter	Description
minLeafSampleCount	Optional. The minimum number of samples per leaf node. The value must be an integer in the range of [100, 1000]. Default value: 500.
groupIDColName	Optional. The name of the stratification column. The whole table is considered as a stratum by default.
sampleRatio	Optional. The fraction of samples collected for training. Valid values: (0, 1). Default value: 0.6.
maxDepth	Optional. The maximum depth of a tree. The value must be an integer in the range of [1, 11]. Default value: 11.
modelName	The name of the output model.
metricType	Optional. The type of a metric. Valid values: 0 and 1. 0 represents normalized discounted cumulative gain (NDCG) and 1 represents discounted cumulative gain (DCG).
featureRatio	Optional. The fraction of features collected for training. Valid values: (0, 1). Default value: 0.6.
inputTablePartitions	Optional. The partitions selected from the input prediction table. If no partitions are specified, the whole table is selected.
testRatio	Optional. The fraction of testing samples. Valid values: [0, 1]. Default value: 0.0.
featureColNames	The names of feature columns selected from the input table for training.
treeCount	Optional. The number of trees. Valid values: [1, 10000]. Default value: 500.

4.4.6.1.2. Linear SVM

Support-vector machines (SVMs) are developed based on the VC dimension theory and the structural risk minimization principle.

This linear SVM version is not implemented using the kernel function. For more information, see Trust Region Method for L2-SVM at http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf. This algorithm only supports binary classification.

Procedure

1. Configure column settings.
   - Feature Columns: You can select a feature column of the bigint or double type.
   - Label Column: The data type of the label column can be bigint, double, or string. This component only supports binary classification.
2. Set parameters.

Parameters

Parameter	Description
Positive Sample Label	Optional. The value of the positive sample. If this parameter is not specified, the system randomly selects a value. We recommend that you specify this parameter when the positive and negative samples are significantly different.
Positive Penalty Factor	Optional. The weight of the positive sample. Valid values: (0, +∞). Default value: 1.0.
Negative Penalty Factor	Optional. The weight of the negative sample. Valid values: (0, +∞). Default value: 1.0.
Convergence Coefficient	Optional. The convergence deviation. Valid values: (0, 1). Default value: 0.001. Note If no base value is specified, Positive Penalty Factor and Negative Penalty Factor must be set to the same value.

3. View the output. For more information, see the description of the Random Forest component.

PAI command (F/L setup settings are not used)

```
PAI -name LinearSVM
-project algo_public
-DnegativeCost="1.0"
-DmodelName="xlab_m_LinearSVM_6143"
-DpositiveCost="1.0"
-Depsilon="0.001"
-DlabelColName="y"
-DfeatureColNames="pdays,emp_var_rate,cons_conf_idx"
-DinputTableName="bank_data"
-DpositiveLabel="0";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
negativeCost	Optional. The weight of the negative sample. It is the penalty factor of the negative sample. Valid values: (0, +∞). Default value: 1.0.

Parameter	Description
modelName	The name of the output model.
positiveCost	Optional. The weight of the positive sample. It is the penalty factor of the positive sample. Valid values: (0, +∞). Default value: 1.0.
epsilon	Optional. The convergence coefficient. Valid values: (0, 1). Default value: 0.001.
labelColName	The name of the label column.
featureColNames	The names of feature columns selected from the input table for training.
inputTableName	The name of the input table for training.
positiveLabel	Optional. The value of the positive sample. If this parameter is not specified, the system randomly selects a value.

4.4.6.1.3. Logistic regression for binary classification

Binary classification is a classic logistic regression method. Logistic regression on the algorithm platform supports multiclass classification. The logistic regression component supports two data types: sparse and dense.

Parameter settings

Parameters

Parameter	Description
Regularization Type	Optional. The type of regularization. Valid values: *L1*, *L2*, and *None*. Default value: *L1*.
Maximum Iterations	Optional. The maximum number of L-BFGS iterations. Default value: 100.
Regularization Coefficient	Optional. The regularization coefficient. Default value: 1.0. If regularizedType is set to *None*, this parameter is ignored.
Minimum Convergence Deviance	Optional. The condition to terminate L-BFGS. This is the log-likelihood deviation between two iterations. Default value: *1.0e-06*.

The logistic regression component outputs a model, which is available in the model list.

Model name format: `Experiment Name + "-" + Component Name + "model"` .

PAI command (F/L setup settings are not used)

```
PAI -name LogisticRegression
-project algo_public
-DmodelName="xlab_m_logistic_regression_6096"
-DregularizedLevel="1"
-DmaxIter="100"
-DregularizedType="l1"
-Depsilon="0.000001"
-DlabelColName="y"
-DfeatureColNames="pdays,emp_var_rate"
-DgoodValue="1"
-DinputTableName="bank_data";
```

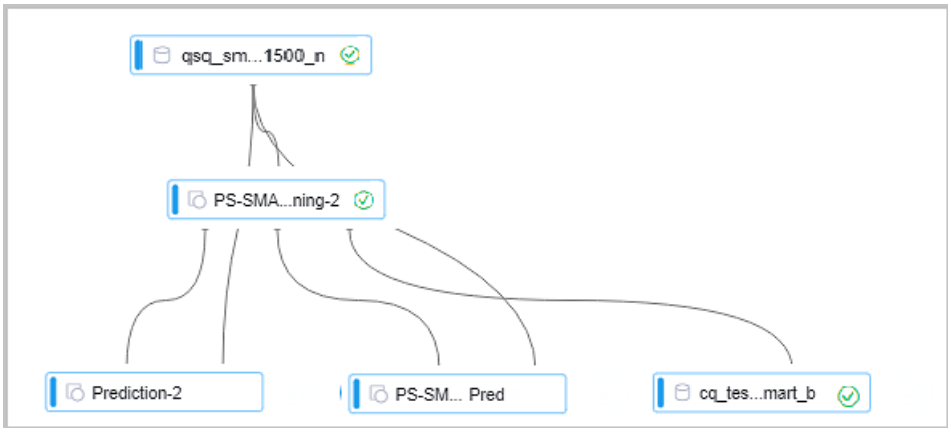
Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
modelName	The name of the output model.
regularizedLevel	Optional. The regularization coefficient. Default value: 1.0. If regularizedType is set to *None*, this parameter is ignored.
maxIter	Optional. The maximum number of L-BFGS iterations. Default value: 100.
regularizedType	Optional. The type of regularization. Valid values: *L1*, *L2*, and *None*. Default value: *L1*.
epsilon	Optional. The convergence deviation. It is the condition to terminate L-BFGS. This is the log-likelihood deviation between two iterations. Default value: *1.0e-06*.
labelColName	The name of the label column selected from the input table.
featureColNames	The names of feature columns selected from the input table for training.
goodValue	Optional. The base value. For binary classification, specify the label value of the training coefficient. If this parameter is not specified, the system randomly selects a value.
inputTableName	The name of the input table for training.

4.4.6.1.4. PS-SMART binary classification

A parameter server (PS) is used to train a large number of models online and offline. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient Boosting Decision Tree (GBDT) on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover for high stability. PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and histogram approximation for training acceleration.

Quick start



As shown in the figure, a PS-SMART binary classification model is learned based on training data. The model has three output ports:

- Output model: offline model, which is connected to the unified prediction component. This model does not support the output of leaf node numbers.
- Output model table: a binary table that is not readable and is used to ensure compatibility with the PS-SMART prediction component. The table supports the output of leaf node numbers, which ensures higher efficiency, less resource consumption, and higher stability.
- Output feature importance table: lists the importance of each feature. Three importance types are supported. For more information, see Parameters.

PAI command

- Training

```
PAI -name ps_smart
    -project algo_public
    -DinputTableName="smart_binary_input"
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545859_2"
    -DoutputImportanceTableName="pai_temp_24515_545859_3"
    -DlabelColName="label"
    -DfeatureColNames="f0,f1,f2,f3,f4,f5"
    -DenableSparse="false"
    -Dobjective="binary:logistic"
    -Dmetric="error"
    -DfeatureImportanceType="gain"
    -DtreeCount="5";
    -DmaxDepth="5"
    -Dshrinkage="0.3"
    -Dl2="1.0"
    -Dl1="0"
    -Dlifecycle="3"
    -DsketchEps="0.03"
    -DsampleRatio="1.0"
    -DfeatureRatio="1.0"
    -DbaseScore="0.5"
    -DminSplitLoss="0"
```

- Prediction

```
PAI -name prediction
    -project algo_public
    -DinputTableName="smart_binary_input";
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545860_1"
    -DfeatureColNames="f0,f1,f2,f3,f4,f5"
    -DappendColNames="label,qid,f0,f1,f2,f3,f4,f5"
    -DenableSparse="false"
    -Dlifecycle="28"
```

Parameters

- Data parameters

Command option	Parameter	Description	Valid values	Remarks
featureColNames	Feature Columns	The names of feature columns selected from the input table for training.	If the column name is in dense format, it must be of the bigint or double type. If the column name is in sparse KV format, it must be a string, and its keys and values must be numeric.	Required
labelColName	Label Column	The name of the label column selected from the input table.	The column name can be of either string or numeric type, but only numeric data can be stored in the columns. For example, in binary classification, the column value can be 0 or 1.	Required
weightCol	Weight Column	This column specifies the weight of each sample.	The column name can be of the numeric type.	Optional. Default value: null.
enableSparse	Use Sparse Format	This parameter specifies whether the data in the input table is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons (:), for example, 1:0.3 3:0.9.	[true, false]	Optional. Default value: false.
inputTableName	Input Table Name	N/A	N/A	Required

Command option	Parameter	Description	Valid values	Remarks
modelName	Output Model Name	N/A	N/A	Required
outputImportanceTableName	Output Feature Importance Table Name	N/A	N/A	Optional. Default value: null.
inputTablePartitions	Input Table Partitions	N/A	N/A	Optional. The parameter value must be in ds=1/pt=1 format.
outputTableName	Output Model Table Name	The output table is a MaxCompute table that uses the binary format and is not readable. The prediction component that comes with SMART can be used to generate leaf node numbers.	String	Optional
lifecycle	Output Table Lifecycle	N/A	Positive integer	Optional. Default value: 3.

- Algorithm parameters

Command option	Parameter	Description	Valid values	Remarks
objective	Objective Function Type	The objective function type affects learning and must be selected properly. Select **binary:logistic** for binary classification.	N/A	Required
metric	Evaluation Indicator Type	Evaluation indicators in the training set, which are exported to stdout of the coordinator in a logview.	logloss, error and auc	Optional. Default value: null.

Command option	Parameter	Description	Valid values	Remarks
treeCount	Trees	The number of trees. The training time is proportional to this number.	Positive integer	Optional. Default value: 1.
maxDepth	Maximum Tree Depth	The maximum depth of a tree. We recommend that you set this value to 5, which means the tree can contain up to 32 leaf nodes.	A positive integer in the range of [1, 20]	Optional. Default value: 5.
sampleRatio	Data Sampling Fraction	The data sampling rate when trees are built. The sample data is used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means data sampling is disabled.
featureRatio	Feature Sampling Fraction	The feature sampling rate when trees are built. The sample features are used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means feature sampling is disabled.
l1	L1 Penalty Coefficient	This parameter determines the number of leaf nodes. The greater the value, the less the leaf nodes. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 0.

Command option	Parameter	Description	Valid values	Remarks
l2	L2 Penalty Coefficient	This parameter determines the size of a leaf node. The greater the value, the more evenly the leaf nodes are distributed. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 1.0.
shrinkage	Learning Rate	N/A	(0, 1]	Optional. Default value: 0.3.
sketchEps	Sketch-based Approximate Precision	The threshold for selecting quantiles when you build a sketch. The number of buckets is O(1.0/sketchEps). The smaller the parameter value, the more buckets are generated. Typically, you do not need to modify this value.	(0, 1)	Optional. Default value: 0.03.
minSplitLoss	Minimum Split Loss Change	The minimum split loss changes required for splitting a node. The greater the value, the more conservatively the node splits.	Non-negative real number	Optional. Default value: 0.

Command option	Parameter	Description	Valid values	Remarks
featureNum	Features	The number of features or the maximum feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional
baseScore	Global Offset	Original predicted values of all samples.	Real number	Optional. Default value: 0.5.
featureImportanceType	Feature Importance Type	The type of feature importance. **weight** indicates the number of times that a feature splits. **gain** indicates information gain brought by the feature. **cover** indicates the number of samples that the feature covers on the splitting nodes.	**weight**, **gain**, and **cover**	Optional. Default value: **gain**.

- Note
  - Specify different values for the objective parameter in different learning models. On the binary classification Web GUI, the objective function is automatically specified and invisible to users. On the command line, set the objective parameter to `binary:logistic`.
  - Mappings between metrics and objective functions are: **logloss** for **negative loglikelihood for logistic regression**, **error** for **binary classification error**, and **auc** for **Area under curve for classification**.

Execution optimization

Command option	Parameter	Description	Valid values	Remarks
coreNum	Cores	The number of cores. The greater the value, the faster the computing algorithm runs.	Positive integer	Optional. Automatically calculated.

Command option	Parameter	Description	Valid values	Remarks
memSizePerCore	Memory Size per Core (MB)	The memory size of each core, where 1024 represents 1 GB of memory.	Positive integer	Optional. Automatically calculated.

Example

- Data generation

  The following example uses data in dense format.

```
drop table if exists lm_test_input;
create table smart_binary_input lifecycle 3 as
select
*
from
(
select 0.72 as f0, 0.42 as f1, 0.55 as f2, -0.09 as f3, 1.79 as f4, -1.2 as f5, 0 as label from dual
union all
select 1.23 as f0, -0.33 as f1, -1.55 as f2, 0.92 as f3, -0.04 as f4, -0.1 as f5, 1 as label from dual
union all
select -0.2 as f0, -0.55 as f1, -1.28 as f2, 0.48 as f3, -1.7 as f4, 1.13 as f5, 1 as label from dual
union all
select 1.24 as f0, -0.68 as f1, 1.82 as f2, 1.57 as f3, 1.18 as f4, 0.2 as f5, 0 as label from dual
union all
select -0.85 as f0, 0.19 as f1, -0.06 as f2, -0.55 as f3, 0.31 as f4, 0.08 as f5, 1 as label from dual
union all
select 0.58 as f0, -1.39 as f1, 0.05 as f2, 2.18 as f3, -0.02 as f4, 1.71 as f5, 0 as label from dual
union all
select -0.48 as f0, 0.79 as f1, 2.52 as f2, -1.19 as f3, 0.9 as f4, -1.04 as f5, 1 as label from dual
union all
select 1.02 as f0, -0.88 as f1, 0.82 as f2, 1.82 as f3, 1.55 as f4, 0.53 as f5, 0 as label from dual
union all
select 1.19 as f0, -1.18 as f1, -1.1 as f2, 2.26 as f3, 1.22 as f4, 0.92 as f5, 0 as label from dual
union all
select -2.78 as f0, 2.33 as f1, 1.18 as f2, -4.5 as f3, -1.31 as f4, -1.8 as f5, 1 as label from dual
) tmp;
```

- Training

  Configure the training data and training components, as shown in Quick start. Select the label column as the target column and columns f0, f1, f2, f3, f4, f5 as feature columns.

- You do not need to set the number of features because this number is calculated automatically by the algorithm. If you have a large number of features and want the algorithm to accurately estimate the amount of required resources, specify the actual number of features.
- To accelerate the training, set the number of cores on the execution optimization page. The greater the number, the faster the algorithm runs. Typically, you do not need to enter the memory size per core because the algorithm can accurately calculate the memory size. The PS algorithm starts to run only when all hosts have obtained the required resources. Therefore, you may need to wait for a longer period of time when the cluster is busy and resources are requested in large volumes.
- You can view the output values of the metrics in the stdout of the coordinator in a logview (HTTP link starting with http://logview.odps.aliyun-inc.com:8080/logview). A single PS-SMART training job can contain multiple tasks, and therefore multiple logviews are created. Select the logview whose name starts with PS to view the output of the PS job.

- Prediction
  - Use the unified prediction component

    The model generated after training is saved in binary format and can be used for prediction. Configure the input model and test data for the prediction component, as shown in Quick start.

    If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.) If the KV format is used, set the data format to sparse format and select the correct delimiter. In the SMART model, key-value pairs are separated by space characters. Therefore, the delimiter must be set to space or \u0020 (escape expression of spaces).

    In the "prediction_detail" column, value 1 indicates a positive sample, and value 0 indicates a negative sample. The values following 0 and 1 indicate the probabilities of the corresponding classes.

- Use the PS-SMART prediction component

  The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction. Configure the input model and test data for the prediction component, as shown in Quick start. Set the required parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than a feature column or a target column. The loss function must be set to binary:logistic.

  The **prediction_score** column lists probabilities of predicted positive samples. A sample is predicted as a positive sample if its score is greater than 0.5. Otherwise, it is predicted as a negative sample. The **leaf_index** column lists the predicted leaf node numbers. Each sample has N numbers, where N is the number of decision trees. Each tree is mapped to a number, which indicates the leaf node number of the sample on this tree.

  > Note
  >
  > - The output model table is a binary table that is not readable and is used to support the PS-SMART prediction component. The table provides outputs such as leaf node numbers and evaluation indicators. However, the output table has strict requirements on data formats, which negatively affects user experience. This component is being continually improved, and may be replaced by another component in the future.
  > - A string type column must be selected as the label column. You can enter strings in the column but cannot be blank or NULL. A feature column can be converted to the string type by using the data type conversion component.
  > - The loss function must be explicitly set to **binary:logistic**. By default, the function does not work.

- View feature importance

  To view feature importance, you can export the third output port to an output table, or right-click **PS-SMART training component** and choose **View Data > Output Feature Importance Table** from the shortcut menu.

order	id	value
1	0	0.5690338015556335
2	1	0.21714292466640472
3	4	0.21382322907447815

  In the table, the ID column lists the numbers of input features. In this example, the data is in dense format. The input features are **f0,f1,f2,f3,f4,f5**. Therefore, ID 0 represents f0 and ID 4 represents f4. If the KV format is used, the IDs represent keys in key-value pairs. Each value indicates a feature importance type. The default value is **gain**, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only three features because only these three features are used during the tree split process. In this case, the importance of unused features is 0.

FAQ

- Q: Does PS_SMART support non-numerical features and tags?
- A: No.

- Q: What is the scale of features supported by PS-SMART? Can we use large-scale 0-1 features?
- A: Although PS-SMART supports tasks that contain hundreds of thousands of features, such tasks consume large amounts of resources and run slowly. Therefore, we recommend that you do not use such a large number of features. The GBDT algorithm is suitable for training with continuous features. The categorical features require one-hot coding to filter out infrequent features before they can be used for training. The continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.
- Q: Why is the result different every time although the SMART algorithm has the same data and the same parameter settings?
- A: The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items introduce data and feature sampling respectively. In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not introduce too much variation to the output model. Therefore, it is normal situation to obtain different results after the algorithm runs multiple times with the same data and same parameter settings.

> Note
>
> - The target column in a PS-SMART binary classification model supports only numerical values (0 for negative samples and 1 for positive samples). Even if values in the MaxCompute table are strings, they are saved as numerical values. If the classification target is a type string similar to **Good** or **Bad**, convert it to 1 or 0.
> - In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are strings, use the serialization component to serialize them. If the feature values are classification type strings, perform feature engineering, such as discretization.

4.4.6.2. Multiclass classification

4.4.6.2.1. KNN

The KNN algorithm is used to resolve classification issues. For a row in the prediction table, this component selects K entries nearest to the row from the training table. It then assigns the row to the class that is most common among the K entries.

PAI command

```
PAI -name knn
    -DtrainTableName=pai_knn_test_input
    -DtrainFeatureColNames=f0,f1
    -DtrainLabelColName=class
    -DpredictTableName=pai_knn_test_input
    -DpredictFeatureColNames=f0,f1
    -DoutputTableName=pai_knn_test_output
    -Dk=2;
```

Parameters

Parameter	Description	Valid value	Default value
trainTableName	Required. The name of the training table.	Table name	N/A
trainFeatureColNames	Required. The names of feature columns selected from the training table.	Column name	N/A
trainLabelColName	Required. The name of the label column selected from the training table.	Column name	N/A
trainTablePartitions	Optional. The partitions selected from the training table.	Partition name	All partitions are selected by default.
predictTableName	Required. The name of the prediction table.	Table name	N/A
outputTableName	Required. The name of the output table.	Table name	N/A
predictFeatureColNames	Optional. The names of feature columns selected from the prediction table.	Column name	N/A
predictTablePartitions	Optional. The partitions selected from the prediction table.	Partition name	All partitions are selected by default.

Parameter	Description	Valid value	Default value
appendColNames	Optional. The names of columns appended to the output table from the prediction table.	Column name	N/A
outputTablePartition	Optional. The partitions in the output table.	Partition name	The output table is non-partitioned by default.
k	Optional. The number of the nearest neighbors.	A positive integer in the range of [1, 1000]	100
enableSparse	Optional. This parameter specifies whether the data in the input table is in sparse format.	true and false	false
itemDelimiter	Optional. The delimiter used to separate key-value pairs when the data in the input table is in sparse format.	Symbol	The default delimiter is a space.
kvDelimiter	Optional. The delimiter used to separate keys and values when the data in the input table is in sparse format.	Symbol	The default delimiter is a colon (:).
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 20000].	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Examples

- **Test data**

```
create table pai_knn_test_input as
select * from
(
 select 1 as f0,2 as f1, 'good' as class from dual
 union all
 select 1 as f0,3 as f1, 'good' as class from dual
 union all
 select 1 as f0,4 as f1, 'bad' as class from dual
 union all
 select 0 as f0,3 as f1, 'good' as class from dual
 union all
 select 0 as f0,4 as f1, 'bad' as class from dual
)tmp;
```

- **PAI command**

```
PAI -name knn
    -DtrainTableName=pai_knn_test_input
    -DtrainFeatureColNames=f0,f1
    -DtrainLabelColName=class
    -DpredictTableName=pai_knn_test_input
    -DpredictFeatureColNames=f0,f1
    -DoutputTableName=pai_knn_test_output
    -Dk=2;
```

- **Output description**

f0	f1	prediction_result	prediction_score	prediction_detail
1	4	bad	1.0	{"bad": 1}
0	4	bad	1.0	{"bad": 1}
0	3	bad	0.5	{ "bad": 0.5, "good": 0.5}
1	3	good	1.0	{"good": 1}
1	2	good	1.0	{"good": 1}

  - f0 and f1: the appended columns in the output table.
  - prediction_result: the classification result.
  - prediction_score: the probabilities for the classification result.
  - prediction_detail: the latest K conclusions and their probabilities.

4.4.6.2.2. Logistic regression for multiclass classification

Logistic regression of Apsara Stack Machine Learning Platform for AI supports multiclass classification. The logistic regression component supports two data formats: sparse and dense.

Parameter settings

Parameters

Parameter	Description
Regularization Type	Optional. The type of regularization. Valid values: *L1*, *L2*, and *None*. Default value: *L1*.
Max Iterations	Optional. The maximum number of L-BFGS iterations. Default value: 100.
Regularization Coefficient	Optional. The regularization coefficient. Default value: 1.0. If regularizedType is set to *None*, this parameter is ignored.
Minimum Convergence Deviance	Optional. The condition to terminate L-BFGS. This is the log-likelihood deviance between two iterations. Default value: *1.0e-06*.

The logistic regression component outputs a model, which is available in the model list.

Model naming format: `Experiment Name + "-" + Component Name + "model"` .

PAI command (F/L setup settings are not used)

```
PAI -name LogisticRegression
-project algo_public
-DmodelName="xlab_m_logistic_regression_6096"
-DregularizedLevel="1"
-DmaxIter="100"
-DregularizedType="l1"
-Depsilon="0.000001"
-DlabelColName="y"
-DfeatureColNames="pdays,emp_var_rate"
-DgoodValue="1"
-DinputTableName="bank_data";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
modelName	The name of the output model.
regularizedLevel	Optional. The regularization coefficient. Default value: 1.0. If regularizedType is set to *None* , this parameter is ignored.

Parameter	Description
maxIter	Optional. The maximum number of L-BFGS iterations. Default value: 100.
regularizedType	Optional. The type of regularization. Valid values: *L1*, *L2*, and *None*. Default value: *L1*.
epsilon	Optional. The convergence deviation. It is the condition to terminate L-BFGS. This is the loglikelihood deviation between two iterations. Default value: *1.0e-06*.
labelColName	The name of the label column selected from the input table.
featureColNames	The names of feature columns selected from the input table for training.
goodValue	Optional. The base value. For multiclass classification, specify the label value of the training coefficient. If this parameter is not specified, the system randomly selects a value.
inputTableName	The name of the input table for training.

4.4.6.2.3. Random forest

A random forest is a classifier that contains multiple decision trees. Its output class is decided by the mode of individual tree output classes.

Procedure

1. Drag and drop the Random Forest component onto the canvas and select columns.

Parameters

Parameter	Description
Feature Columns	Optional. All columns except the label and weight columns are selected by default.
Excluded Columns	Optional. This parameter is used to exclude specified columns from training. This parameter is mutually exclusive with featureColNames.
Columns Forced to Convert	Optional. The default feature parsing rules are as follows: ○ Parse columns of string, boolean, and datetime types to discrete columns. ○ Parse columns of double and bigint types to contiguous columns. ○ Set the *forceCategorical* parameter to parse bigint type columns to categorical columns.
Weight Columns	Optional. You can select all columns except the input and label columns. The double and bigint types are supported.
Label Column	You can select all columns except the input column. The bigint, double, and string types are supported.

2. Set the parameters of the Random Forest component.

Parameters

Parameter	Description
Trees	The number of trees in the forest. Valid values: (0, 1000).
Single-tree Algorithm Type	Optional. The algorithm type of each tree in the forest. Valid values: id3, c4.5, and cart. If the forest has n trees and the condition is `algorithmTypes = a,b`, then `[0,a)` indicates `id3`, `[a,b)` indicates `cart`, and `[b,n)` indicates `c4.5`. If this parameter is set to [2, 4] for a forest with five trees, [0, 1) indicates the ID3 algorithm, [2, 3) indicates the CART algorithm, and 4 indicates the C4.5 algorithm. If the value is None, the algorithms are evenly allocated across the forest.
Random Features per Tree	The number of features selected randomly. Valid values: 1 to N. N indicates the number of features.
Minimum Samples per Leaf Node	Optional. The minimum number of samples per leaf node. The value must be a positive integer no less than 2.
Minimum Fraction of Samples on Leaf Node to Samples on Parent Node	The minimum fraction of samples on a leaf node to samples on a parent node. A value of -1 indicates that no limit is set. Default value: -1. Valid values: [0, 1].
Maximum Tree Depth	The maximum depth of a tree. -1 indicates a completely grown tree. Valid values: [1, ∞).
Random Samples Input per Tree	The number of random samples input per tree. Valid values: (1000, 1000000].

Note

- With improvement of the bagging method, the Random Forest component builds a forest without correlated trees in the big cube. Random forests are similar to the boosting method in many aspects, particularly their training processes.
- For the growth of a single tree, this method provides the id3, cart, and c4.5 options. The treeNum parameter is used to specify the number of trees in the forest, in the range of [1, 1000]. The structure of a single tree can be controlled based on the edited template. You can use other parameters to specify the minimum number of samples per leaf node, the minimum fraction of samples on a leaf node to samples on a parent node, and the maximum depth of a tree.
- Each row in the weight column corresponds to a sample and indicates the proportion of this sample in training. If the age column is selected as the weight column, the sample in the row with a higher weight value in the age column has a higher proportion during the training.
- The "input table is empty!" error may occur in the following situations: The sampling fraction is too small, which means that the value of maxRecordSize is too small, or the input table is empty.

PAI command (F/L setup settings are not used)

```
PAI -name RandomForests
-project algo_public
-DmodelName="xlab_m_random_forests_6036"
-DrandomColNum="1.0"
-DlabelColName="campaign"
-DmaxTreeDeep="10"
-DmaxRecordSize="100000"
-DfeatureColNames="age,pdays,previous,emp_var_rate,cons_price_idx,cons_conf_idx,euribor3m,nr_em
ployed"
-DisFeatureContinuous="1,1,1,1,1,1,1,1"
-DminNumPer="-1"
-DminNumObj="2"
-DinputTableName="bank_data"
-DweightColName="y"
-DtreeNum="10";
```

Parameters

Parameter	Description
name	The name of the component.

Parameter	Description
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
modelName	The name of the output model.
randomColNum	Optional. The random attribute type. This parameter specifies the number of features randomly selected each time a single tree is generated. *-1* indicates `log2N`. Valid values: 1 to N. N indicates the number of features.
labelColName	The name of the label column selected from the input table.
maxTreeDeep	Optional. The maximum depth of a tree. *-1* indicates a completely grown tree. Valid values: [1, ∞].
maxRecordSize	Optional. The maximum number of samples per tree. Valid values: (1000, 1000000). *-1* indicates 100000.
featureColNames	The names of feature columns selected from the input table for training.
isFeatureContinuous	Specifies whether the feature for subsequent columns is continuous or discrete. *1* indicates that the feature column data is continuous, while *0* indicates that the feature column data is discrete. *1,0,0* indicates that values are continuous in the first feature column and discrete in the second and third feature columns. The number of values corresponds to the feature length.
minNumPer	Optional. The minimum fraction of samples on a leaf node to samples on a parent node. A value of *-1* indicates that no limit is set. Valid values: [0.0, 1.0].
minNumObj	Optional. The minimum number of samples per leaf node.
inputTableName	The name of the input table for training.
weightColName	Optional. The name of the weight column selected from the input table. If there is no weight column, set this parameter to *None*. If there is any weight column, the value of weightColName is greater than 0.
treeNum	The number of trees. Valid values: (0, 1000).

4.4.6.2.4. Naive Bayes

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong independence assumptions between the features. A probabilistic model that can more accurately describe this potential is called an independent feature model.

Component description

Component parameter settings (For more information, see the description of the Random Forest component.)

PAI command

```
PAI -name NaiveBayes
-project algo_public
-DmodelName="xlab_m_NaiveBayes_23772"
-DinputTablePartitions="pt=20150501"
-DlabelColName="poutcome"
-DfeatureColNames="age,previous,cons_conf_idx,euribor3m"
-DisFeatureContinuous="1,1,1,1"
-DinputTableName="bank_data_partition";
```

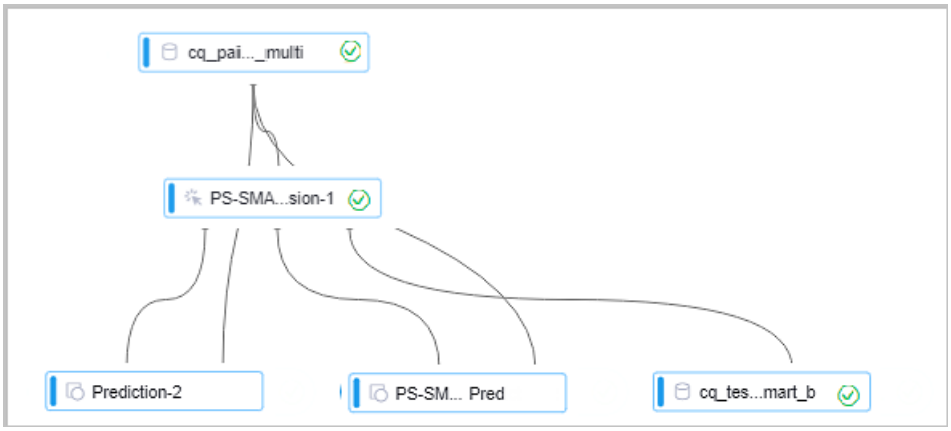
Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
modelName	The name of the model generated by training.
inputTablePartitions	Optional. The partitions selected from the input prediction table. If no partitions are specified, the entire table is selected.
labelColName	The name of the label column selected from the input table.
featureColNames	The names of feature columns selected from the input table for training.
isFeatureContinuous	Specifies whether the feature for subsequent columns is continuous or discrete. *1* indicates that the feature column data is continuous, while *0* indicates that the feature column data is discrete. *1,0,0* indicates that values are continuous in the first feature column and discrete in the second and third feature columns. The number of values corresponds to the feature length.
inputTableName	The name of the input table.

4.4.6.2.5. PS-SMART multiclass classification

A parameter server (PS) is used to train a large number of models online and offline. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient Boosting Decision Tree (GBDT) on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover for high stability. PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and histogram approximation for training acceleration.

Quick start



As shown in the figure, a PS-SMART multiclass classification model is learned based on training data. The model has three output ports:

- Output model: offline model, which is connected to the unified prediction component. This model does not support the output of leaf node numbers.
- Output model table: a binary table that is not readable and is used to support the PS-SMART prediction component. The table provides outputs such as leaf node numbers and assessment metrics. However, the output table has strict requirements on data formats, which negatively affects user experience. This component is being continually improved, and may be replaced by another component in the future.
- Output feature importance table: lists the importance of each feature. Three importance types are supported. For more information, see Parameters.

PAI command

- Training

```
PAI -name ps_smart
    -project algo_public
    -DinputTableName="smart_multiclass_input"
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545859_2"
    -DoutputImportanceTableName="pai_temp_24515_545859_3"
    -DlabelColName="label"
    -DfeatureColNames="features"
    -DenableSparse="true"
    -Dobjective="multi:softprob"
    -Dmetric="mlogloss"
    -DfeatureImportanceType="gain"
    -DtreeCount="5";
    -DmaxDepth="5"
    -Dshrinkage="0.3"
    -Dl2="1.0"
    -Dl1="0"
    -Dlifecycle="3"
    -DsketchEps="0.03"
    -DsampleRatio="1.0"
    -DfeatureRatio="1.0"
    -DbaseScore="0.5"
    -DminSplitLoss="0"
```

- Prediction

```
PAI -name prediction
    -project algo_public
    -DinputTableName="smart_multiclass_input";
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545860_1"
    -DfeatureColNames="features"
    -DappendColNames="label,features"
    -DenableSparse="true"
    -DkvDelimiter=":"
    -Dlifecycle="28"
```

Parameters

- Data parameters

Command option	Parameter	Description	Valid values	Remarks
featureColNames	Feature Column	The names of feature columns selected from the input table for training.	If the column name is in dense format, it must be of the bigint or double type. If the column name is in sparse KV format, it must be a string, and its keys and values must be numeric.	Required
labelColName	Label Column	The name of the label column selected from the input table.	The column name can be of either string or numeric type, but only numeric data can be stored in the columns. For multiclass classification, column values can be 0, 1, 2, ..., n-1, where n is the number of classes.	Required
weightCol	Weight Column	This column specifies the weight of each sample.	The column name can be of the numeric type.	Optional. Default value: null.
enableSparse	Use Sparse Format	This parameter specifies whether the data in the input table is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons (:), for example, 1:0.3 3:0.9.	true, false	Optional. Default value: false.
inputTableName	Input Table Name	N/A	N/A	Required

Command option	Parameter	Description	Valid values	Remarks
modelName	Output Model Name	N/A	N/A	Required
outputImportanceTableName	Output Feature Importance Table Name	N/A	N/A	Optional. Default value: null.
inputTablePartitions	Input Table Partitions	N/A	N/A	Optional. The parameter value must be in ds=1/pt=1 format.
outputTableName	Output Model Table	The output table is a MaxCompute table that uses the binary format and is not readable. The prediction component that comes with SMART can be used to generate leaf node numbers.	String	Optional
lifecycle	Output Table Lifecycle	N/A	Positive integer	Optional. Default value: 3.

- Algorithm parameters

Command option	Parameter	Description	Valid values	Remarks
classNum	Classes	The number of classes in multiclass classification. If the number of classes is n, the label column name can be 0, 1, 2, ..., or n-1.	A non-negative integer, greater than or equal to 3.	Required

Command option	Parameter	Description	Valid values	Remarks
objective	Objective Function Type	The objective function type affects learning and must be selected properly. Set it to **multi:softprob** for multiclass classification.	N/A	Required
metric	Evaluation Indicator Type	Evaluation indicators in the training set, which are exported to stdout of the coordinator in a logview.	**mloglossmerror**	Optional. Default value: null.
treeCount	Trees	The number of trees. The training time is proportional to this number.	Positive integer	Optional. Default value: 1.
maxDepth	Maximum Decision Tree Depth	The maximum depth of a tree. We recommend that you set this value to 5, which means the tree can contain up to 32 leaf nodes.	A positive integer in the range of [1, 20].	Optional. Default value: 5.
sampleRatio	Data Sampling Fraction	The data sampling rate when trees are built. The sample data is used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means data sampling is disabled.

Command option	Parameter	Description	Valid values	Remarks
featureRatio	Feature Sampling Fraction	The feature sampling rate when trees are built. The sample features are used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means feature sampling is disabled.
l1	L1 Penalty Coefficient	This parameter determines the number of leaf nodes. The greater the value, the fewer the leaf nodes. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 0.
l2	L2 Penalty Coefficient	This parameter determines the size of a leaf node. The greater the value, the more evenly the leaf nodes are distributed. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 1.0.
shrinkage	Learning Rate	N/A	(0, 1]	Optional. Default value: 0.3.

Command option	Parameter	Description	Valid values	Remarks
sketchEps	Sketch-based Approximate Precision	The threshold for selecting quantiles when you build a sketch. The number of buckets is O(1.0/sketchEps). The smaller the parameter value, the more buckets are generated. Typically, you do not need to modify this value.	(0, 1)	Optional. Default value: 0.03.
minSplitLoss	Minimum Split Loss	The minimum split loss changes required for splitting a node. The greater the value, the more conservatively the node splits.	Non-negative real number	Optional. Default value: 0.
featureNum	Features	The number of features or the maximum feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional
baseScore	Global Offset	Original predicted values of all samples.	Real number	Optional. Default value: 0.5.

Command option	Parameter	Description	Valid values	Remarks
featureImportanceType	Feature Importance Type	The type of feature importance. **weight** indicates the number of times that a feature splits. **gain** indicates information gain brought by the feature. **cover** indicates the number of samples that the feature covers on the splitting nodes.	**weight**, **gain**, and **cover**	Optional. Default value: **gain**.

- Note
  - Specify different values for the objective parameter in different learning models. On the multiclass classification Web GUI, the objective function is automatically specified and invisible to users. On the command line, set the objective parameter to `multi:softprob`.
  - Mappings between metrics and objective functions are: **mlogloss** for **multiclass negative log likelihood**, and **merror** for **multiclass classification error**.
- Execution optimization

Command option	Parameter	Description	Valid values	Remarks
coreNum	Cores	The number of cores. The greater the value, the faster the computing algorithm runs.	Positive integer	Optional. Automatically calculated.
memSizePerCore	Memory Size per Core (MB)	The memory size of each core, where 1024 represents 1 GB of memory.	Positive integer	Optional. Automatically calculated.

Example

- Data generation

  The following example uses data in sparse KV format.

```
drop table if exists smart_multiclass_input;
create table smart_multiclass_input lifecycle 3 as
select
*
from
(
select 2 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
   union all
select 1 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
   union all
select 1 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
   union all
select 2 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
   union all
select 1 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
   union all
select 1 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
   union all
select 0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
   union all
select 1 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
   union all
select 0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
   union all
select 1 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

The data has five dimensions of features.

- Training

  Configure the training data and training components, as shown in Quick start. Select the label column as the target column and the features column as the feature column.

  - You do not need to set the number of features because this number is calculated automatically by the algorithm. If you have a large number of features and want the algorithm to accurately estimate required resources, specify the actual number of features.
  - To accelerate the training, set the number of cores on the execution optimization page. The greater the number, the faster the algorithm runs. Typically, you do not need to enter the memory size per core because the algorithm can accurately calculate the memory size. The PS algorithm starts to run only when all hosts have obtained the required resources. Therefore, you may need to wait for a longer time when the cluster is busy and resources are requested in large volumes.

- You can view the output values of the metrics in the stdout of the coordinator in a logview (HTTP link starting with http://logview.odps.aliyun-inc.com:8080/logview). A single PS-SMART training job can contain multiple tasks, which creates multiple logviews. Select the logview whose name starts with PS to view the output of the PS job.

Then, perform operations in the logview.

- Prediction
  - Use the unified prediction component

    The model generated after training is saved in binary format and can be used for prediction. Configure the input model and test data for the prediction component, as shown in Quick start.

    If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.) If the KV format is used, set the data format to sparse format and select the correct delimiter. In the SMART model, key-value pairs are separated with spaces. Therefore, the delimiter must be set to space or \u0020 (escape expression of spaces).

    In the **prediction_detail**column, values 0, 1, and 2 indicate classes, and the values following them indicate probabilities of the corresponding classes. The **predict_result** column lists the selected classes with the highest probability, and the **predict_score** column lists the probability of each selected class.

  - Use the PS-SMART prediction component

    The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction. Configure the input model and test data for the prediction component, as shown in Quick start. Set the required parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than a feature column or a target column. The loss function must be explicitly set to **multi:softprob**.

    The **score_class_k** columns list probabilities of class k. The class with the highest probability is the predicted class. The **leaf_index** column lists the predicted leaf node numbers. Each sample has N × M numbers, where N is the number of decision trees, and M is the number of classes. In this example, each sample has 15 numbers (5 × 3 = 15). Each tree is mapped to a number, which indicates the leaf node number of the sample on this tree.

    > **Note**
    > - The output model table is a binary table that is not readable and is used to support the PS-SMART prediction component. The table provides outputs such as leaf node numbers and evaluation indicators. However, the output table has strict requirements on data formats, which negatively affects user experience. This component is being continually improved, and may be replaced by another component in the future.
    > - A string type column must be selected as the label column. You can enter strings in the column but cannot be blank or NULL. A feature column can be converted to the string type by using the data type conversion component.
    > - The loss function must be explicitly set to **multi:softprob**. By default, the loss function does not work.

- View feature importance

  To view feature importance, you can export the third output port to an output table, or right-click **PS-SMART training component** and choose **View Data > Output Feature Importance Table** from the shortcut menu.The following figure shows the output feature importance table.

order	id	value
1	1	0.276059627532959
2	3	0.20854459702968597
3	4	0.31002077460289
4	5	0.20537501573562622

  In the table, the ID column lists the numbers of input features. In this example, the data is in KV format, and the IDs represent keys in key-value pairs. If the dense format is used and input features are **f0,f1,f2,f3,f4,f5**, ID 0 represents f0 and ID 4 represents f4. Each value indicates a feature importance type. The default value is **gain**, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only four features because only these four features are used during the tree split process. In this case, the importance of unused features is 0.

FAQ

- Q: Does PS_SMART support non-numerical features and tags?
- A: No.
- Q: What is the scale of features supported by PS-SMART? Can we use large-scale 0-1 features?
- A: Although PS-SMART supports tasks that contain hundreds of thousands of features, such tasks consume large amounts of resources and run slowly. Therefore, we recommend that you do not use a large number of features. The GBDT algorithm is suitable for training with continuous features. The categorical features require one-hot coding (to filter out infrequent features) before they can be used for training. The continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.
- Q: Why is the result different every time although the SMART algorithm has the same data and the same parameter settings?
- A: The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items introduce data and feature sampling respectively. In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not introduce too much variation to the output model. Therefore, it is normal situation to obtain different results after the algorithm runs multiple times with the same data and same parameter settings.

Note

- The target column in a PS-SMART multiclass classification model supports only positive integer IDs (class numbers are 0, 1, 2, ..., n-1, where n is the number of classes). Even if the values in the MaxCompute table are strings, they are saved as numerical values. If the classification target is a type string similar to **Good**, **Medium**, or **Bad**, convert it into a numeric value (0, 1, 2, ..., n-1).
- In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are strings, use the serialization component to serialize them. If the feature values are classification type strings, perform feature engineering, such as discretization.

4.4.6.3. K-means clustering

K-means clustering is a widely used algorithm that is used to divide n objects into k clusters while maintaining high similarity within each cluster. Similarity is calculated based on the average value of objects in a cluster. This algorithm is similar to the expectation maximization algorithm for calculating mixed normality distribution, as both algorithms try to find the natural clustering center in data. K-means clustering randomly selects k objects. Each object represents the average value or center of a cluster. Based on its distance from each cluster center, each remaining object is then assigned to the nearest cluster and the average value of each cluster is re-calculated. This process is repeated until the criterion function converges. This algorithm assumes that object properties are from the spatial vector. Its objective is to minimize the sum of the mean square deviance inside each group.

Parameter settings

Parameters

Parameter	Description
Clusters	The number of clusters. Default value: 10.
Distance Measurement Method	Valid values: *euclidean*, *cityblock* (the sum of absolute deviations), and *cosine*. Default value: euclidean.
Initial Centroid Location	Valid values: *sample* (randomly selected), *topk* (first K rows), *uniform* (evenly distributed and randomly generated), *matrix* (an initial centroid table must be specified), and *kmpp* (k-means++ initialization). Default value: sample.
Maximum Iterations	The maximum number of iterations. Default value: 100.
Minimum Iteration Precision	The minimum iteration precision. Default value: 0.0.

Procedure

1. After running the K-means Clustering component, you can view the cluster center table.

Cluster center table: The number of columns in this table is equal to the total number of columns selected from the input table. The number of rows is equal to the number of clusters, with each row representing a cluster center location.

2. Right-click the target table and choose **View Data** to view the cluster index table (idxTablename).
   - Cluster index table: The number of rows is equal to the total number of rows in the input table. The value in each row represents the cluster index of the point in the corresponding row of the input table.
   - The names of all columns are displayed. A classification marking column is appended to the table.
   - 0, 1, 2, 3 are classification IDs.
   - You can also use the table name generated by PAI command to view the cluster center table, cluster index table, and cluster count table in IDE.

Note If *matrix* is selected as the initial centroid location, you must define the initial centroid table, with the same columns as the original table. The number of rows is the same as the number of clusters. When you prepare the table, configure k centers and use SQL or MapReduce for sampling, or select another method based on your requirements.

PAI command

```
PAI -name KMeans
-project algo_public
-DcenterCount="10"
-DidxTableName="bank_data_index"
-DdistanceType="euclidean"
-DappendColsIndex="0,1,2,3,4,5,6,7,8,9,10"
-DcenterTableName="pai_temp_3300_27298_3"
-Dloop="100"
-DclusterCountTableName="pai_temp_3300_27298_2"
-DinitCentersMethod="sample"
-Daccuracy="0.0"
-DinputTableName="bank_data"
-DselectedColNames="cons_conf_idx,emp_var_rate,euribor3m,pdays,previous";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.

Parameter	Description
centerCount	The number of clusters. The value must be an integer. Default value: 10.
idxTableName	The name of the output cluster index table. The number of rows is equal to the total number of rows in the input table. The value in each row represents the cluster index of the point in the corresponding row of the input table.
distanceType	Optional. The method used to measure the distance. Valid values: euclidean, cityblock, and cosine. Default value: euclidean.
appendColsIndex	Optional. The name of the ID column appended to the output table. No ID column is appended to the output table by default.
centerTableName	The name of the output cluster center table. The number of columns in this table is equal to the total number of columns selected from the input table. The number of rows is equal to the number of clusters, with each row representing a cluster center location.
loop	Optional. The maximum number of iterations. The value must be an integer. Default value: 100.
clusterCountTableName	The name of the cluster point count table. The number of rows is equal to the number of clusters, which indicates the total number of cluster points in the class in each clustering centroid row.
initCentersMethod	Optional. The method used to determine the initial centroid location. The options include sample (randomly selected), topk (first K rows), uniform (evenly distributed and randomly generated), matrix (an initial centroid table must be specified), and kmpp (k-means++ initialization). Default value: sample.
accuracy	The minimum iteration precision. Default value: 0.0.
inputTableName	The name of the input table.
selectedColNames	The names of columns selected from the input table, which are separated with commas (,). Only double type is supported.
initCenterTableName	Optional. The name of the table that stores the initial center values. This table is not required unless initCentersMethod is set to *matrix*.

4.4.6.4. Regression

4.4.6.4.1. GBDT regression

Gradient boosting decision tree (GBDT) is an iterative decision tree algorithm based on multiple decision trees. The final output is the sum of conclusions of all trees. GBDT can be applied to almost all regression models (linear or nonlinear) and has a wider scope of application than logistic regression that is only applicable to linear regression.

For more information, see A Regression Framework for Learning Ranking Functions Using Relative Relevance Judgments. For more information, see GBDT binary classification.

PAI command

```
PAI -name gbdt
-project algo_public
-DfeatureSplitValueMaxSize="500"
-DlossType="0"
-DrandSeed="0"
-DnewtonStep="0"
-Dshrinkage="0.05"
-DmaxLeafCount="32"
-DlabelColName="campaign"
-DinputTableName="bank_data_partition"
-DminLeafSampleCount="500"
-DsampleRatio="0.6"
-DgroupIDColName="age"
-DmaxDepth="11"
-DmodelName="xlab_m_GBDT_83602"
-DmetricType="2"
-DfeatureRatio="0.6"
-DinputTablePartitions="pt=20150501"
-Dtau="0.6"
-Dp="1"
-DtestRatio="0.0"
-DfeatureColNames="previous,cons_conf_idx,euribor3m"
-DtreeCount="500"
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
featureColNames	Optional. The names of feature columns selected from the input table for training.	Column name	All columns are selected by default.

Parameter	Description	Valid values	Default value
labelColName	Optional. The name of the label column selected from the input table.	Column name	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions are selected by default.
modelName	Required. The name of the output model.	-	-
outputImportanceTableName	Optional. The name of the output feature importance table.	-	-
groupIDColName	Optional. The name of the stratification column.	Column name	The whole table is selected by default.
lossType	Optional. The loss function type. The function types include *0: GBRANK*, *1:LAMBDAMART_DCG*, *2: LAMBDAMART_NDCG*, *3: LEAST_SQUARE*, and *4: LOG_LIKELIHOOD*.	0, 1, 2, 3, and 4	0
metricType	Optional. The type of metrics. *0(NDCG)* indicates the normalized discounted cumulative gain, *1 (DCG)* indicates the discounted cumulative gain, and *2 (AUC)* is applicable only to 0/1 label.	0, 1, and 2	2

Parameter	Description	Valid values	Default value
treeCount	Optional. The number of trees.	[1, 10000]	500
shrinkage	Optional. The learning rate.	(0, 1]	0.05
maxLeafCount	Optional. The maximum number of leaves. This value must be an integer.	[2, 1000]	32
maxDepth	Optional. The maximum depth of a tree. This value must be an integer.	[1, 11]	11
minLeafSampleCount	Optional. The minimum number of samples on a leaf node. This value must be an integer.	[100, 1000]	500
sampleRatio	Optional. The fraction of training samples.	(0, 1]	0.6
featureRatio	Optional. The fraction of training features.	(0, 1]	0.6
tau	Optional. The Tau parameter in gbrank loss.	[0, 1]	0.6
p	Optional. The p parameter in gbrank loss.	[1, 10]	1
randSeed	Optional. The random seed.	[0, 10]	0
newtonStep	Optional. This parameter specifies whether to use the Newton method.	0 and 1	1
featureSplitValueMaxSize	Optional. The maximum number of splits per feature.	[1, 1000]	500
lifecycle	Optional. The lifecycle of the output table.	-	No lifecycle is set by default.

Examples

SQL statement to generate data:

```
drop table if exists gbdt_ls_test_input;
create table gbdt_ls_test_input as select     * from (
select        cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label    from dual    union all
select          cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label    from dual    union all
select          cast(0 as double) as f0,
cast(0 as double) as f1,
cast(1 as double) as f2,
cast(0 as double) as f3,
cast(1 as bigint) as label    from dual    union all
select          cast(0 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(1 as double) as f3,
cast(1 as bigint) as label    from dual    union all
select          cast(1 as double) as f0,
cast(0 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label    from dual    union all
select          cast(0 as double) as f0,
cast(1 as double) as f1,
cast(0 as double) as f2,
cast(0 as double) as f3,
cast(0 as bigint) as label    from dual ) a;
```

PAI command

- Training:

```
drop offlinemodel if exists gbdt_ls_test_model;
PAI -name gbdt
-project algo_public
-DfeatureSplitValueMaxSize="500"
-DlossType="3"
-DrandSeed="0"
-DnewtonStep="1"
-Dshrinkage="0.5"
-DmaxLeafCount="32"
-DlabelColName="label"
-DinputTableName="gbdt_ls_test_input"
-DminLeafSampleCount="1"
-DsampleRatio="1"
-DmaxDepth="10"
-DmetricType="0"
-DmodelName="gbdt_ls_test_model"
-DfeatureRatio="1"
-Dp="1"
-Dtau="0.6"
-DtestRatio="0"
-DfeatureColNames="f0,f1,f2,f3"
-DtreeCount="10"
```

- Prediction:

```
drop table if exists gbdt_ls_test_prediction_result;
PAI -name prediction
-project algo_public
-DdetailColName="prediction_detail"
-DmodelName="gbdt_ls_test_model"
-DitemDelimiter=","
-DresultColName="prediction_result"
-Dlifecycle="28"
-DoutputTableName="gbdt_ls_test_prediction_result"
-DscoreColName="prediction_score"
-DkvDelimiter=":"
-DinputTableName="gbdt_ls_test_input"
-DenableSparse="false"
-DappendColNames="label"
```

Input description

gbdt_ls_test_input

f0	f1	f2	f3	label
1.0	0.0	0.0	0.0	0
0.0	0.0	1.0	0.0	1
0.0	0.0	0.0	1.0	1
0.0	1.0	0.0	0.0	0
1.0	0.0	0.0	0.0	0
0.0	1.0	0.0	0.0	0

Output description

gbdt_ls_test_prediction_result

label	prediction_result	prediction_score	prediction_detail
0	NULL	0.0	{"label": 0}
0	NULL	0.0	{"label": 0}
1	NULL	0.9990234375	{"label": 0.9990234375}
1	NULL	0.9990234375	{"label": 0.9990234375}
0	NULL	0.0	{"label": 0}
0	NULL	0.0	{"label": 0}

4.4.6.4.2. Linear regression

This component is used to resolve regression issues and analyze the linear relationship between a dependent variable and multiple independent variables. Certain columns from an input table are selected as feature columns and one column is selected as the label column for linear regression training and linear regression model generation.

PAI command

```
PAI -name linearregression
    -project algo_public
    -DinputTableName=lm_test_input
    -DfeatureColNames=x
    -DlabelColName=y
    -DmodelName=lm_test_input_model_out;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
modelName	Required. The name of the output model.	-	-
outputTableName	Optional. The name of the output model evaluation table.	This parameter must be specified when enableFitGoodness is set to *true*.	-
labelColName	Required. The name of the label column.	The name must be a double or bigint type value. Only one column can be specified.	-
featureColNames	Required. The name of the feature column.	The name must be a double or bigint type value in dense format, or a string type value in sparse format. Multiple columns can be specified.	-
inputTablePartitions	Optional. The partitions selected from the input table for training.	-	No partitions are selected by default.
maxIter	Optional. The maximum number of iterations.	-	100
epsilon	Optional. The minimum likelihood deviance.	-	0.000001

Parameter	Description	Valid values	Default value
enableSparse	Optional. This parameter specifies whether the data is in sparse format.	true and false	false
enableFitGoodness	Optional. This parameter specifies whether to perform model evaluation. Model evaluation can be performed using a variety of metrics, including R-squared, adjusted R-Squared, Akaike information criterion, degrees of freedom, residual standard deviation, and deviation.	true and false	false
enableCoefficientEstimate	Optional. This parameter specifies whether to estimate the regression coefficient. The metrics of this parameter are value t, value p, and confidence interval [2.5%, 97.5%]. This parameter takes effect only when enableFitGoodness is set to *true*. This parameter is ignored when enableFitGoodness is set to *false*.	true and false	false
itemDelimiter	Optional. The delimiter used to separate key-value pairs. This parameter takes effect only when enableSparse is set to *true*.	-	Use spaces on command lines and use commas (,) on webpages.

Parameter	Description	Valid values	Default value
kvDelimiter	Optional. The delimiter used to separate keys and values. This parameter takes effect only when enableSparse is set to *true*.	-	The default delimiter is a colon (:).
lifecycle	Optional. The lifecycle of the output table.	An integer greater than or equal to -1	Default value: -1. This value indicates that no lifecycle is set.
coreNum	Optional. The number of cores.	An integer larger than 0	Default value: -1. This value indicates that the number of instances is determined by the amount of input data.
memSizePerCore	Optional. The memory size of each core.	(100, 65536)	Default value: -1. This value indicates that the memory size is determined by the amount of input data.

Examples

- SQL statement to generate data:

```
drop table if exists lm_test_input;
    create table lm_test_input as
    select
     *
    from
    (
     select 10 as y, 1.84 as x1, 1 as x2, '0:1.84 1:1' as sparsecol1 from dual
      union all
     select 20 as y, 2.13 as x1, 0 as x2, '0:2.13' as sparsecol1 from dual
      union all
     select 30 as y, 3.89 as x1, 0 as x2, '0:3.89' as sparsecol1 from dual
      union all
     select 40 as y, 4.19 as x1, 0 as x2, '0:4.19' as sparsecol1 from dual
      union all
     select 50 as y, 5.76 as x1, 0 as x2, '0:5.76' as sparsecol1 from dual
      union all
     select 60 as y, 6.68 as x1, 2 as x2, '0:6.68 1:2' as sparsecol1 from dual
      union all
     select 70 as y, 7.58 as x1, 0 as x2, '0:7.58' as sparsecol1 from dual
      union all
     select 80 as y, 8.01 as x1, 0 as x2, '0:8.01' as sparsecol1 from dual
      union all
     select 90 as y, 9.02 as x1, 3 as x2, '0:9.02 1:3' as sparsecol1 from dual
      union all
     select 100 as y, 10.56 as x1, 0 as x2, '0:10.56' as sparsecol1 from dual
    ) tmp;
```

- PAI command

```
PAI -name linearregression
    -project algo_public
    -DinputTableName=lm_test_input
    -DlabelColName=y
    -DfeatureColNames=x1,x2
    -DmodelName=lm_test_input_model_out
    -DoutputTableName=lm_test_input_conf_out
    -DenableCoefficientEstimate=true
    -DenableFitGoodness=true
    -Dlifecycle=1;
pai -name prediction
    -project algo_public
    -DmodelName=lm_test_input_model_out
    -DinputTableName=lm_test_input
    -DoutputTableName=lm_test_input_predict_out
    -DappendColNames=y;
```

- Output description:
  - When enableFitGoodness is set to *true*, partitions specified by `p='goodness'` are created in the model evaluation table. The output metrics are *R-squared*, *adjusted R-Squared*, *Akaike information criterion*, *degrees of freedom*, *residual standard deviation*, and *deviation*.
  - When enableCoefficientEstimate is set to *true*, partitions specified by `p='coefficient'` are created in the model evaluation table. The table contains the intercepts and the *name*, *coefficient*, *t-score*, *p-value*, and *confidence interval [2.5%, 97.5%]* of the features.

- Output model evaluation table: lm_test_input_conf_out.

colname	value	tscore	pvalue	confidenceinterval	p
Intercept	-6.42378496687763	-2.2725755951390028	0.06	{"2.5%": -11.964027, "97.5%": -0.883543}	coefficient
x1	10.260063429838898	23.270944360826963	0.0	{"2.5%": 9.395908, "97.5%": 11.124219}	coefficient
x2	0.35374498323846265	0.2949247320997519	0.81	{"2.5%": -1.997160, "97.5%": 2.704650}	coefficient
rsquared	0.9879675667384592	NULL	NULL	NULL	goodness
adjusted_rsquared	0.9845297286637332	NULL	NULL	NULL	goodness
aic	59.331109494251805	NULL	NULL	NULL	goodness
degree_of_freedom	7.0	NULL	NULL	NULL	goodness
standardErr_residual	3.765777749448906	NULL	NULL	NULL	goodness
deviance	99.267574407711128	NULL	NULL	NULL	goodness

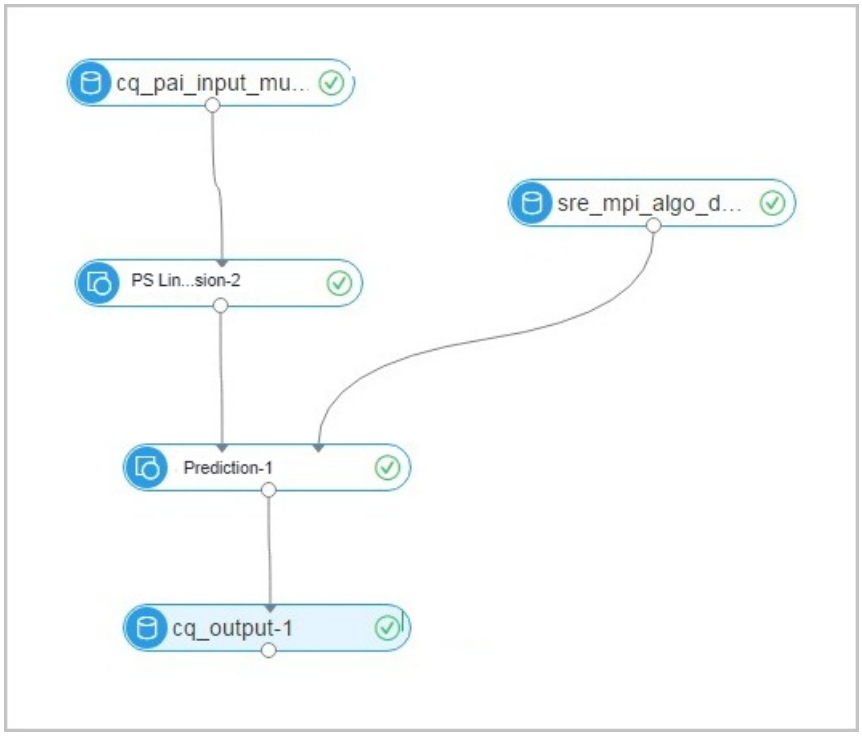
- Output prediction table: lm_test_input_predict_out.

y	prediction_result	prediction_score	prediction_detail
10	NULL	12.808476727264404	{"y": 12.8084767272644}
20	NULL	15.43015013867922	{"y": 15.43015013867922}
30	NULL	33.48786177519568	{"y": 33.48786177519568}
40	NULL	36.565880804147355	{"y": 36.56588080414735}
50	NULL	52.674180388994415	{"y": 52.67418038899442}
60	NULL	62.82092871092313	{"y": 62.82092871092313}
70	NULL	71.34749583130122	{"y": 71.34749583130122}
80	NULL	75.75932310613193	{"y": 75.75932310613193}
90	NULL	87.1832221199846	{"y": 87.18322211998461}
100	NULL	101.92248485222113	{"y": 101.9224848522211}

4.4.6.4.3. PS linear regression

Linear regression is a classic regression algorithm used to analyze the linear relationship between a dependent variable and multiple independent variables. Parameter servers (PSs) are used to run large amounts of training tasks online and offline. Parameter servers can use hundreds of billions of samples to efficiently train billions of feature models. The PS linear regression model can run training tasks with hundreds of billions of samples and billions of features, and supports L1 and L2 regular expressions.

Quick start



PAI command

- **Training**

```
PAI -name ps_linearregression
    -project algo_public
    -DinputTableName="lm_test_input"
    -DmodelName="linear_regression_model"
    -DlabelColName="label"
    -DfeatureColNames="features"
    -Dl1Weight=1.0
    -Dl2Weight=0.0
    -DmaxIter=100
    -Depsilon=1e-6
    -DenableSparse=true
```

- **Prediction**

```
drop table if exists logistic_regression_predict;
PAI -name prediction
    -DmodelName="linear_regression_model"
    -DoutputTableName="linear_regression_predict"
    -DinputTableName="lm_test_input"
    -DappendColNames="label,features"
    -DfeatureColNames="features"
    -DenableSparse=true
```

Parameters

- **Data parameters**

Command option	Parameter	Description	Valid values	Default value
featureColNames	Feature Columns	Required. The names of feature columns selected from the input table for training.	If a column name is in dense format, it must be of the bigint or double type. If the column name is in sparse KV format, it must be a string.	-
labelColName	Label Column	Required. The name of the label column selected from the input table.	The column name must be of the bigint or double type.	-
enableSparse	Use Sparse Format	Optional. If you choose to use the sparse KV format, do not use feature ID 0. We recommend that the feature IDs start from 1.	true and false	false
itemDelimiter	KV Pair Delimiter	Optional. The delimiter used to separate key-value pairs when data in the input table is in sparse format.	Symbol	The default delimiter is a space.

Command option	Parameter	Description	Valid values	Default value
kvDelimiter	KV Delimiter	Optional. The delimiter used to separate keys and values when data in the input table is in sparse format.	Symbol	The default delimiter is a colon (:).
inputTableName	Input Table Name	Required.	Table name	-
modelName	Output Model Name	Required.	Model name	-
inputTablePartitions	Input Table Partitions	Optional.	Partition name	The parameter value must be in the ds=1/pt=1 format.
enableModelIo	Output to Offline Model	Optional. When this parameter is set to false, the data is output to a MaxCompute table where you can view model weights.	true and false	true

- **Algorithm parameters**

Command option	Parameter	Description	Valid values	Default value
l1Weight	L1 Weight	Optional. The L1 regularization coefficient. The larger this value is, the fewer non-zero elements a model has. To overfit the model, set this parameter to a larger value.	A non-negative real number	1.0

Command option	Parameter	Description	Valid values	Default value
l2Weight	L2 Weight	Optional. The L2 regularization coefficient. The larger this value is, the smaller the absolute values of the model parameters are. To overfit the model, set this parameter to a larger value.	A non-negative real number	0
maxIter	Maximum Iterations	Optional. The maximum number of LBFGS/OWL-QN iterations. Value 0 indicates that no limit is set.	A non-negative integer	100
epsilon	Minimum Convergence Deviance	Optional. The mean of the relative loss change rates in ten iterations, which is used as a condition to determine whether to terminate the optimization algorithm. The smaller this value is, the stricter the condition is, and the longer the algorithm runs.	A real number between 0 and 1	1.0e-06

Command option	Parameter	Description	Valid values	Default value
modelSize	Largest Feature ID	Optional. The largest feature ID among all feature IDs (feature dimension). It can be larger than the actual largest feature ID. The larger this value is, the higher the memory usage is. If you leave this parameter empty, the system starts an SQL task to calculate the largest feature ID automatically.	A non-negative integer	0

Note Both the maximum iterations and minimum convergence deviance determine when the algorithm stops. If both parameters are set, the algorithm stops when one of the conditions is met.

- **Execution optimization**

Command option	Parameter	Description	Valid values	Default value
coreNum	Cores	Optional. The number of cores. The larger this value is, the faster the computing algorithm runs.	A positive integer	Automatically allocated.
memSizePerCore	Memory Size per Core (MB)	Optional. The memory size of each core, where 1024 represents 1 GB of memory.	A positive integer	Automatically allocated. Typically, you do not need to set this parameter because the algorithm can accurately estimate the memory size required.

Examples

- **Data generation**

  The following example uses data in sparse KV format:

```
drop table if exists lm_test_input;
create table lm_test_input as
select
*
from
(
select 2 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
   union all
select 1 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
   union all
select 1 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
   union all
select 2 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
   union all
select 1 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
   union all
select 1 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
   union all
select 0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
   union all
select 1 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
   union all
select 0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
   union all
select 1 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

  The feature IDs start from 1, and the maximum feature ID is 5.

- **Training**

  Configure the training data and training components based on Quick start. Select the label column as the target column and features column as the feature column. Then, select the sparse data format.

  - You can retain the default value 0 for the largest feature ID. The algorithm can start an SQL task to calculate the largest feature ID automatically. If you do not want to start the SQL task, enter a value greater than 5. This value indicates the number of feature columns in dense format and indicates the largest feature ID in KV format.

  - To accelerate the training, you can set the number of cores on the tuning page. The larger the number is, the faster the algorithm runs. Typically, you do not need to enter the memory size per core because the algorithm can accurately calculate the memory size. The PS algorithm starts to run only when all hosts have obtained the resources. Therefore, you may need to wait a longer period of time when the cluster is busy and resources are requested in large volume.

- **Prediction**

  The model generated after training is saved in binary format and can be used for prediction. Configure the input settings (model and testing data) for the prediction component and set parameters based on Quick start.

  Select the KV format for training and set a correct delimiter. When the KV format is used, key-value pairs are separated by spaces. Therefore, the delimiter must be set to space or `\u0020` (the escape expression of space).

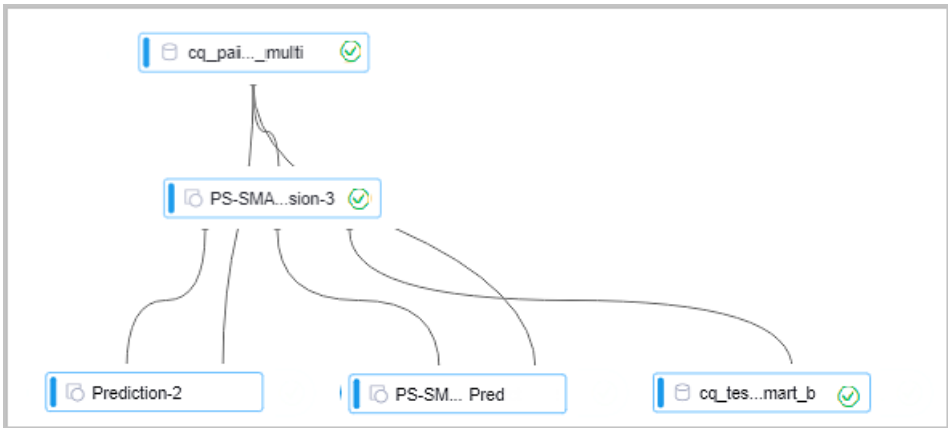
Restrictions and guidelines

In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are strings, use the serialization component to serialize them. If the feature values are classification type strings, perform feature engineering, such as discretization.

4.4.6.4.4. PS-SMART regression

A parameter server (PS) is used to train a large number of models online and offline. Scalable Multiple Additive Regression Tree (SMART) is an implementation of Gradient boosting decesion tree （GBDT） on PS. PS-SMART can run training tasks containing up to tens of billions of samples and hundreds of thousands of features on thousands of nodes. It also supports failover for high stability. PS-SMART supports various data formats, training targets, evaluation targets, output feature importance, and histogram approximation for training acceleration.

Quick start



As shown in the figure, a PS-SMART regression model is learned based on training data. The model has three output ports:

- Output model: offline model, which is connected to the unified prediction component. This model does not support the output of leaf node numbers.
- Output model table: a binary table that is not readable and is used to ensure compatibility with the PS-SMART prediction component. The table provides outputs such as leaf node

numbers and evaluation metrics. However, the output table has strict requirements on data formats, which negatively affects user experience. This component is being continually improved, and may be replaced by another component in the future.

- Output feature importance table: lists the importance of each feature. Three importance types are supported. For more information, see Parameters.

PAI command

- Training

```
PAI -name ps_smart
    -project algo_public
    -DinputTableName="smart_regression_input"
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545859_2"
    -DoutputImportanceTableName="pai_temp_24515_545859_3"
    -DlabelColName="label"
    -DfeatureColNames="features"
    -DenableSparse="true"
    -Dobjective="reg:linear"
    -Dmetric="rmse"
    -DfeatureImportanceType="gain"
    -DtreeCount="5";
    -DmaxDepth="5"
    -Dshrinkage="0.3"
    -Dl2="1.0"
    -Dl1="0"
    -Dlifecycle="3"
    -DsketchEps="0.03"
    -DsampleRatio="1.0"
    -DfeatureRatio="1.0"
    -DbaseScore="0.5"
    -DminSplitLoss="0"
```

- Prediction

```
PAI -name prediction
    -project algo_public
    -DinputTableName="smart_regression_input";
    -DmodelName="xlab_m_pai_ps_smart_bi_545859_v0"
    -DoutputTableName="pai_temp_24515_545860_1"
    -DfeatureColNames="features"
    -DappendColNames="label,features"
    -DenableSparse="true"
    -Dlifecycle="28"
```

Parameters

- Data parameters

Command option	Parameter	Description	Valid values	Remarks
featureColNames	Feature Column	The names of feature columns selected from the input table for training.	If the column name is in dense format, it must be of the bigint or double type. If the column name is in sparse KV format, it must be a string, and its keys and values must be numeric.	Required
labelColName	Label Column	The name of the label column selected from the input table.	The column name can be of either string or numeric type, but only numeric data can be stored in the columns. For example, the column value can be 0 or 1 for regression.	Required
weightCol	Weight Column	This column specifies the weight of each sample.	The column name can be of the numeric type.	Optional. Default value: null.

Command option	Parameter	Description	Valid values	Remarks
enableSparse	Use Sparse Format	This parameter specifies whether the data in the input table is in sparse format, in which key-value pairs are separated by spaces whereas keys and values are separated by colons (:), for example, 1:0.3 3:0.9.	true, false	Optional. Default value: false.
inputTableName	Input Table Name	N/A	N/A	Required
modelName	Output Model Name	N/A	N/A	Required
outputImportanceTableName	Output Feature Importance Table Name	N/A	N/A	Optional. Default value: null.
inputTablePartitions	Input Table Partitions	N/A	N/A	Optional. The parameter value must be in ds=1/pt=1 format.
outputTableName	Output Model Table Name	The output table is a MaxCompute table that uses the binary format and is not readable. The prediction component that comes with SMART can be used to generate leaf node numbers.	String	Optional
lifecycle	Output Table Lifecycle	N/A	Positive integer	Optional. Default value: 3.

- Algorithm parameters

Command option	Parameter	Description	Valid values	Remarks
objective	Objective Function Type	The objective function type affects learning and must be selected properly. Multiple loss functions are available for regression. For more information, see the notes.		Required. The default type is Linear regression.
metric	Evaluation Indicator Type	Evaluation indicators in the training set, which must correspond to the objective function type and are exported to stdout of the coordinator in a logview. For more information, see the following notes and samples.		Optional. Default value: null.
treeCount	Trees	The number of trees. The training time is proportional to this number.	Positive integer	Optional. Default value: 1.
maxDepth	Maximum Decision Tree Depth	The maximum depth of a tree. We recommend that you set this value to 5, which means the tree can contain up to 32 leaf nodes.	A positive integer in the range of [1, 20]	Optional. Default value: 5.

Command option	Parameter	Description	Valid values	Remarks
sampleRatio	Data Sampling Fraction	The data sampling rate when trees are built. The sample data is used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means data sampling is disabled.
featureRatio	Feature Sampling Fraction	The feature sampling rate when trees are built. The sample features are used to build a weak learner to accelerate training.	(0, 1]	Optional. The default value is 1.0, which means feature sampling is disabled.
l1	L1 Penalty Coefficient	This parameter determines the number of leaf nodes. The greater the value, the fewer the leaf nodes. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 0.
l2	L2 Penalty Coefficient	This parameter determines the size of a leaf node. The greater the value, the more evenly the leaf nodes are distributed. You can set this parameter to a greater value if overfitting occurs.	Non-negative real number	Optional. Default value: 1.0.
shrinkage	Learning Rate	N/A	(0, 1]	Optional. Default value: 0.3.

Command option	Parameter	Description	Valid values	Remarks
sketchEps	Sketch-based Approximate Precision	The threshold for selecting quantiles when you build a sketch. The number of buckets is O(1.0/sketchEps). The smaller the parameter value, the more buckets are generated. Typically, you do not need to change this value.	(0, 1)	Optional. Default value: 0.03.
minSplitLoss	Minimum Split Loss	The minimum split loss changes required for splitting a node. The greater the value, the more conservatively the node splits.	Non-negative real number	Optional. Default value: 0.
featureNum	Features	The number of features or the maximum feature ID. Specify this parameter for resource usage estimation.	Positive integer	Optional
baseScore	Global Offset	Original predicted values of all samples.	Real number	Optional. Default value: 0.5.

Command option	Parameter	Description	Valid values	Remarks
featureImportanceType	Feature Importance Type	The type of feature importance. **weight** indicates the number of times that a feature splits. **gain** indicates information gain brought by the feature. **cover** indicates the number of samples that the feature covers on the splitting nodes.	**weight**, **gain**, and **cover**	Optional. Default value: **gain**.
tweedieVarPower	Tweedie Distribution Index	Tweedie distribution index indicates the relationship between the variance and mean. For example, $Var(y) ~ E(y)^tweedie_variance_power$.	(1, 2)	Optional. Default value: 1.5.

- Note
  - Specify different values for the objective parameter in different learning models. The regression Web GUI provides multiple objective functions.

    reg:linear (Linear regression) // The range of label numbers is (-∞, +∞).
    reg:logistic (Logistic regression) // The range of label numbers is [0, 1].
    count:poisson (Poisson regression for count data, output mean of poisson distribution) // Label numbers must be greater than 0.
    reg:gamma (Gamma regression for modeling insurance claims severity, or for any outcome that might be [gamma-distributed](https://en.wikipedia.org/wiki/Gamma_distribution#Applications)) // Label numbers must be greater than 0.
    reg:tweedie (Tweedie regression for modeling total loss in insurance, or for any outcome that might be [Tweedie-distributed](https://en.wikipedia.org/wiki/Tweedie_distribution#Applications).) // Label numbers must be greater than or equal to 0.

- Metrics for these objective functions are:

  rmse (rooted mean square error, corresponding to objective reg:linear)
  mae (mean absolute error, corresponding to objective reg:linear)
  poisson-nloglik (negative loglikelihood for poisson regression, corresponding to objective count:poisson)
  gamma-deviance (Residual deviance for gamma regression, corresponding to objective reg:gamma)
  gamma-nloglik (Negative log-likelihood for gamma regression, corresponding to objective reg:gamma)
  tweedie-nloglik (tweedie-nloglik@1.5, negative log-likelihood for Tweedie regression, at a specified value of the tweedie_variance_power parameter)

- Execution optimization

Command option	Parameter	Description	Valid values	Remarks
coreNum	Cores	The number of cores. The greater the value, the faster the computing algorithm runs.	Positive integer	Optional. Automatically allocated.
memSizePerCore	Memory Size per Core (MB)	The memory size of each core, where 1024 represents 1 GB of memory.	Positive integer	Optional. Automatically allocated.

Example

- Data generation

  The following example uses data in sparse KV format.

```
drop table if exists smart_regression_input;
create table smart_regression_input as
select
*
from
(
select 2.0 as label, '1:0.55 2:-0.15 3:0.82 4:-0.99 5:0.17' as features from dual
   union all
select 1.0 as label, '1:-1.26 2:1.36 3:-0.13 4:-2.82 5:-0.41' as features from dual
   union all
select 1.0 as label, '1:-0.77 2:0.91 3:-0.23 4:-4.46 5:0.91' as features from dual
   union all
select 2.0 as label, '1:0.86 2:-0.22 3:-0.46 4:0.08 5:-0.60' as features from dual
   union all
select 1.0 as label, '1:-0.76 2:0.89 3:1.02 4:-0.78 5:-0.86' as features from dual
   union all
select 1.0 as label, '1:2.22 2:-0.46 3:0.49 4:0.31 5:-1.84' as features from dual
   union all
select 0.0 as label, '1:-1.21 2:0.09 3:0.23 4:2.04 5:0.30' as features from dual
   union all
select 1.0 as label, '1:2.17 2:-0.45 3:-1.22 4:-0.48 5:-1.41' as features from dual
   union all
select 0.0 as label, '1:-0.40 2:0.63 3:0.56 4:0.74 5:-1.44' as features from dual
   union all
select 1.0 as label, '1:0.17 2:0.49 3:-1.50 4:-2.20 5:-0.35' as features from dual
) tmp;
```

Feature IDs are numbered starting from 1, and the maximum feature ID is 5.

- Training

  Select the label column as the target column and the features column as the feature column.

  - You do not need to set the number of features because this number is calculated automatically by the algorithm. If you have a large number of features and want the algorithm to accurately estimate the amount of required resources, specify the actual number of features.
  - To accelerate the training, set the number of cores on the execution optimization page. The greater the number, the faster the algorithm runs. Typically, you do not need to enter the memory size per core because the algorithm can accurately calculate the memory size. The PS algorithm starts to run only when all hosts have obtained the required resources. Therefore, you may need to wait for a longer time when the cluster is busy and resources are requested in large volumes.

- You can view the output values of the metrics in the stdout of the coordinator in a logview (HTTP link starting with http://logview.odps.aliyun-inc.com:8080/logview). A single PS-SMART training job can contain multiple tasks, and therefore multiple logviews are created. Select the logview whose name starts with PS to view the output of the PS job.

- Prediction
  - Use the unified prediction component

    The model generated after training is saved in binary format and can be used for prediction. Configure the input model and test data for the prediction component, as shown in Quick start.

    If the dense format is used, you only need to select feature columns. (All columns are selected by default, and extra columns do not affect the prediction.) If the KV format is used, set the data format to sparse format and select the correct delimiter. In the SMART model, key-value pairs are separated with spaces. Therefore, the delimiter must be set to space or \u0020 (escape expression of spaces).

  - Use the PS-SMART prediction component

    The output model table obtained after training is saved in binary format and can be used by the PS-SMART prediction component for prediction. Configure the input model and test data for the prediction component, as shown in Quick start. Set the required parameters, including the data format, feature columns, target column, and number of classes. The ID column can only be a string type column other than a feature column or a target column. The loss function must be explicitly set to the objective function used for training.

    The **prediction_score** column lists the predicted values. The **leaf_index** column lists the predicted leaf node numbers. Each sample has N numbers, where N is the number of decision trees. Each tree is mapped to a number, which indicates the leaf node number of the sample on this tree.

    > **Note**
    > - The output model table is a binary table that is not readable and is used to support the PS-SMART prediction component. The table provides outputs such as leaf node numbers and evaluation indicators. However, the output table has strict requirements on data formats, which negatively affects user experience. This component is being continually improved, and may be replaced by another component in the future.
    > - A string type column must be selected as the label column. You can enter strings in the column but cannot be blank or NULL. A feature column can be converted to the string type by using the data type conversion component.
    > - The loss function must be explicitly set to the objective function used for training. By default, the loss function does not work.

- View feature importance

To view feature importance, you can export the third output port to an output table, or right-click **PS-SMART training component** and choose **View Data > Output Feature Importance Table** from the shortcut menu.

order	id	value
1	1	0.14059734344482422
2	4	0.8594027161598206

In the table, the ID column lists the numbers of input features. In this example, the data is in KV format and the IDs represent keys in key-value pairs. If the dense format is used and input features are **f0,f1,f2,f3,f4,f5**, ID 0 represents f0, and ID 4 represents f4. Each value indicates a feature importance type. The default value is **gain**, indicating the sum of information gains brought by a feature in the model. The preceding figure shows only two features because only these two features are used during the tree split process. In this case, the importance of unused features is 0.

FAQ

- Q: Does PS_SMART support non-numerical features and tags?
- A: No.
- Q: What is the scale of features supported by PS-SMART? Can we use large-scale 0-1 features?
- A: Although PS-SMART supports tasks that contain hundreds of thousands of features, such tasks consume large amounts of resources and run slowly. Therefore, we recommend that you do not use a large number of features. The GBDT algorithm is suitable for training with continuous features. The categorical features require one-hot coding (to filter out infrequent features) before they can be used for training. The continuous numerical features can be used for training with the GBDT algorithm directly. Discretization is not recommended for numerical features.
- Q: Why is the result different every time although the SMART algorithm has the same data and the same parameter settings?
- A: The PS-SMART algorithm applies randomness in many scenarios. For example, the data_sample_ratio and fea_sample_ratio items introduce data and feature sampling respectively. In addition, the PS-SMART algorithm uses histograms to show similarity. When multiple workers run in a cluster in distributed mode, local sketches are merged to global sketches in a random order. Although different merging orders result in different tree structures, this does not introduce too much variation to the output model. Therefore, it is normal situation to obtain different results after the algorithm runs multiple times with the same data and same parameter settings.

> **Note**
>
> - The target column in a PS-SMART regression model supports only numerical values. Even if values in the MaxCompute table are strings, they are saved as numerical values.
> - In the key-value format, feature IDs must be positive integers, and feature values must be real numbers. If feature IDs are strings, use the serialization component to serialize them. If feature values are classification type strings, perform feature engineering, such as discretization.

4.4.6.5. Collaborative filtering (etrec)

etrec is an item-based collaborative filtering algorithm that takes two input columns and provides the top N items that have the highest similarity.

Set the user and item columns.

- You can configure three similarity types.
- topN indicates the first N items with the highest similarity.
- Calculation method: the method used to calculate items that appear multiple times.

PAI command

```
PAI -name pai_etrec
-project algo_public
-DsimilarityType="wbcosine"
-Dweight="1"
-DminUserBehavior="2"
-Dlifecycle="28"
-DtopN="2000"
-Dalpha="0.5"
-DoutputTableName="etrec_test_result"
-DmaxUserBehavior="500"
-DinputTableName="etrec_test_input"
-Doperator="add"
-DuserColName="user"
-DitemColName="item"
```

Parameters

Parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	N/A	N/A
userColName	Required. The name of the input table column selected as the user column.	N/A	N/A
itemColName	The name of the input table column selected as the item column.	N/A	N/A

Parameter	Description	Valid value	Default value
payloadColName	Optional. The name of the input table column selected as the payload column.	N/A	No payload column is set by default.
inputTablePartitions	Optional. The partitions selected from the input table for training.	N/A	The whole table is selected by default.
outputTableName	Required. The name of the output table.	N/A	N/A
outputTablePartition	Optional. The partitions in the output table.	N/A	The output table is non-partitioned by default.
similarityType	Optional. The type of similarity.	wbcosine, asymcosine, and jaccard	wbcosine
topN	Optional. N items with the highest similarity.	[1, 10000]	2000
minUserBehavior	Optional. The minimum user behavior.	[2,)	2
maxUserBehavior	Optional. The maximum user behavior.	[2, 100000]	500
itemDelimiter	Optional. The delimiter used to separate items in the output table.	N/A	The default delimiter is a space.
kvDelimiter	Optional. The delimiter used to separate keys and values in the output table.	N/A	The default delimiter is a colon (:).
alpha	Optional. The value of the smoothing factor for asymcosine.	N/A	0.5
weight	Optional. The weight used for asymcosine.	N/A	1.0
operator	Optional. The action to be performed when the same items exist for one user.	add, mul, min, and max	add

Parameter	Description	Valid value	Default value
lifecycle	Optional. The lifecycle of the output table.	N/A	1

Examples

- SQL statement to generate data:

```
drop table if exists etrec_test_input;
create table etrec_test_input as select * from
(
select cast(0 as string) as user,
cast(0 as string) as item from dual
union all
select cast(0 as string) as user,
cast(1 as string) as item from dual
union all
select cast(1 as string) as user,
cast(0 as string) as item from dual
union all
select cast(1 as string) as user,
cast(1 as string) as item from dual ) a;
```

- PAI command

```
drop table if exists etrec_test_result;
PAI -name pai_etrec
-project algo_public
-DsimilarityType="wbcosine"
-Dweight="1"
-DminUserBehavior="2"
-Dlifecycle="28"
-DtopN="2000"
-Dalpha="0.5"
-DoutputTableName="etrec_test_result"
-DmaxUserBehavior="500"
-DinputTableName="etrec_test_input"
-Doperator="add"
-DuserColName="user"
-DitemColName="item"
```

- Input description

etrec_test_input

User	Item
0	0
0	1
1	0
1	1

- Output description

etrec_test_result

Item ID	Similarity
0	1:1
1	0:1

4.4.6.6. Evaluation

4.4.6.6.1. Regression model evaluation

You can evaluate a regression model based on the predicted and actual results. Indicators include SST, SSE, SSR, R2, R, MSE, RMSE, MAE, MAD, MAPE, count, yMean, and predictMean.

PAI command

```
Pai -name regression_evaluation
-project algo_public
-DinputTableName=input_table
-DyColName=y_col
-DpredictionColName=prediction_col
-DoutputTableName=output_table;
```

Parameters

Parameter	Description	Default value
inputTableName	Required. The name of the input table.	-
inputTablePartitions	Optional. The partitions selected from the input table for training.	All partitions of the input table are selected by default.

Parameter	Description	Default value
yColName	Required. The name of the original dependent variable column in the input table. It must be a numerical value.	-
predictionColName	Required. The name of the predicted dependent variable column. It must be a numerical value.	-
outputTableName	Required. The name of the output table.	-
inputTablePartitions	Optional. The partitions selected from the input table.	-
lifecycle	Optional. The lifecycle of the output table.	No lifecycle is set by default.

Output

The following table describes the JSON columns.

Column description

Column	Description
SST	The sum of squares total.
SSE	The sum of squares error.
SSR	The sum of squares regression.
R2	The coefficient of determination.
R	The coefficient of multiple correlations.
MSE	The mean squared error.
RMSE	The root-mean-square error.
MAE	The mean absolute error.
MAD	The mean absolute difference.
MAPE	The mean absolute percentage error.
count	The number of rows.
yMean	The mean of original dependent variables.
predictionMean	The mean of prediction results.

4.4.6.6.2. Clustering model evaluation

You can evaluate clustering models, including metrics and icons, based on raw data and clustering models.

PAI command

```
PAI -name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

Parameters

Parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	Table name	N/A
selectedColNames	Optional. The names of columns selected from the input table for evaluation. The column names must be separated with commas (,). The column names must be the same as the feature names saved in the model.	Column name	All columns are selected by default.
inputTablePartitions	Optional. The partitions selected from the input table for evaluation, in the `name1=value1/name2=value2` format. Separate multiple partitions with commas (,).	N/A	All partitions are selected by default.
enableSparse	Optional. This parameter specifies whether data in the input table is in sparse format.	true and false	false

Parameter	Description	Valid value	Default value
itemDelimiter	Optional. The delimiter used to separate key-value pairs when data in the input table is in sparse format.	N/A	The default delimiter is a space.
kvDelimiter	Optional. The delimiter used to separate keys and values when data in the input table is in sparse format.	N/A	The default delimiter is a colon (:).
modelName	Required. The name of the input clustering model.	Model name	N/A
outputTableName	Required. The name of the output table.	Table name	N/A
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Evaluation formula

The Calinski-Harabasz metric is also known as the variance ratio criterion (VRC), which is defined as follows:

$$VRC_k = \frac{SS_B}{SS_W} \times \frac{(N-k)}{(k-1)},$$

- SS_B represents the inter-clustering variance. The definition is as follows:

  $$SS_B = \sum_{i=1}^{k} n_i \|m_i - m\|^2,$$

  - k represents the number of cluster centers.
  - m_i represents the center of cluster i.
  - m represents the mean of the input data.

- SS_W represents the intra-clustering variance. The definition is as follows:

  $$SS_W = \sum_{i=1}^{k} \sum_{x \in c_i} \|x - m_i\|^2,$$

  - k represents the number of cluster centers.
  - x represents a data point.
  - c_i represents the number i cluster.

- m_i represents the center of cluster i.

- `N` represents the total number of records. `k` represents the number of cluster centers.

Examples

- Test data

```
create table if not exists pai_cluster_evaluation_test_input
as select * from ( select 1 as id,
1 as f0,2 as f3 from dual union all
select 2 as id, 1 as f0,3 as f3 from dual union all
select 3 as id, 1 as f0,4 as f3 from dual union all
select 4 as id, 0 as f0,3 as f3 from dual union all
select 5 as id, 0 as f0,4 as f3 from dual )tmp;
```

- Clustering model building

```
pai -name kmeans
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DcenterCount=3
-Dloop=10
-Daccuracy=0.00001
-DdistanceType=euclidean
-DinitCenterMethod=random
-Dseed=1
-DmodelName=pai_kmeans_test_model
-DidxTableName=pai_kmeans_test_idx
```

- PAI command

```
PAI -name cluster_evaluation
-project algo_public
-DinputTableName=pai_cluster_evaluation_test_input
-DselectedColNames=f0,f3
-DmodelName=pai_kmeans_test_model
-DoutputTableName=pai_ft_cluster_evaluation_out;
```

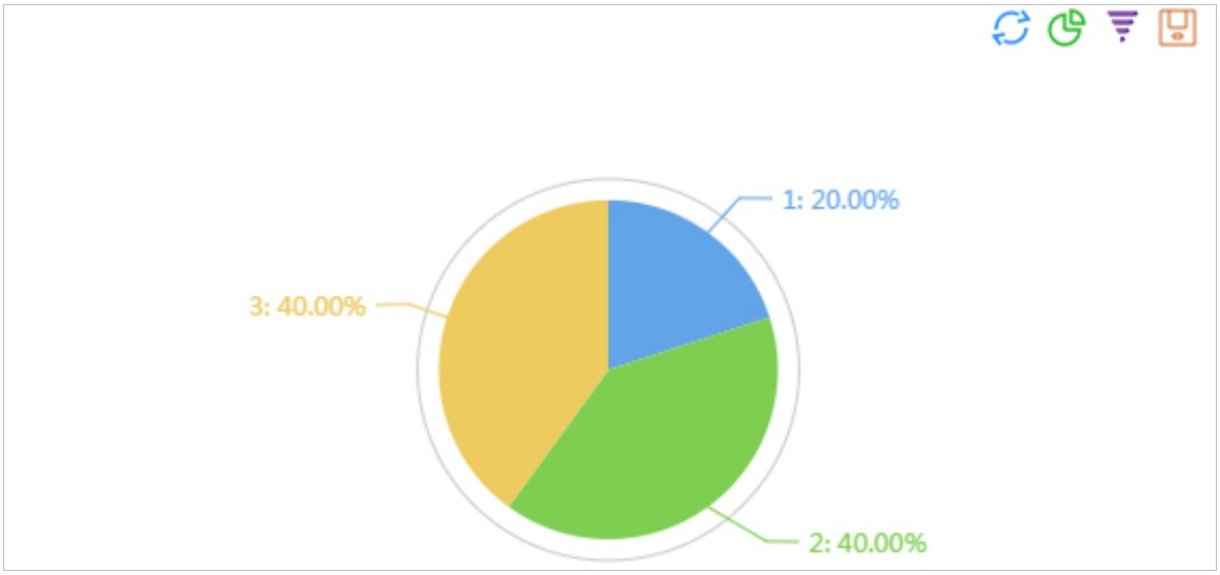
- Output description

Output table (outputTableName)

Column	Description
count	The total number of records.
centerCount	The number of cluster centers.
calinhara	The Calinski Harabasz metric.
clusterCounts	The number of points included in each cluster.

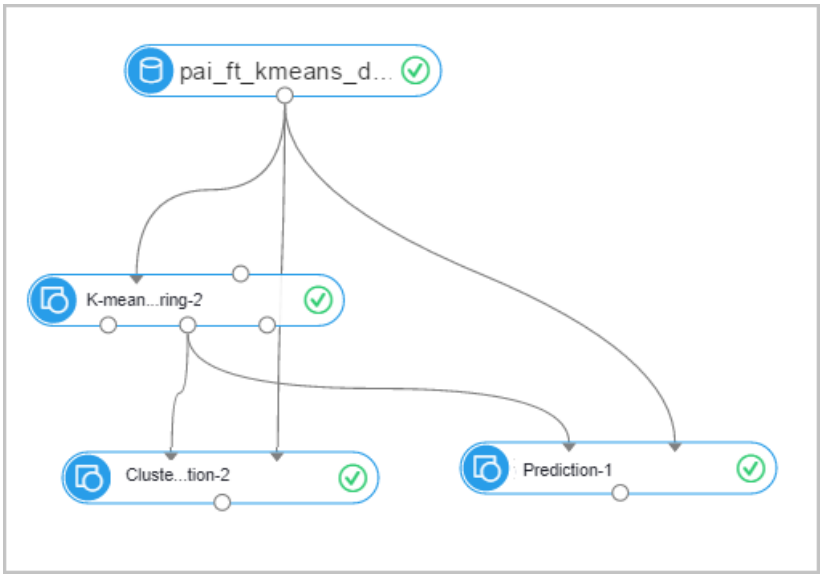
PaiWeb demonstration

Clustering model evaluation



PaiWeb-Pipeline

PaiWeb-Pipeline



4.4.6.6.3. Binary classification evaluation

You can evaluate a regression algorithm model based on its predicted and actual results. The metrics include MSE, MAE, and MAPE.

PAI command

```
pai -name evaluation
-DinputTableName=input_table
-DlabelColName=label_name
-DpredictionColName=prediction_score
-DoutputTableName=output_table;
```

Algorithm parameters

Parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	N/A	N/A
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	N/A	All partitions of the input table are selected by default.
labelColName	Required. The name of the original label column in the input table. It must be a numerical value.	N/A	N/A
predictionColName	Required. The name of the label column in the prediction result table. It must be a numerical value.	N/A	N/A
outputTableName	Required. The name of the output table.	N/A	N/A
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Output table

Output column description

Column	Description
MSE	The mean square error, which is used to measure the mean error and evaluate data changes. The smaller the MSE, the more accurately a prediction model describes the test data.

Column	Description
MAE	The mean absolute error, which is used to measure the mean difference between the predicted value and the actual value.
MAPE	The mean absolute percentage error, which is used to measure prediction accuracy. The value is expressed in percentage. If MAPE is set to 15, the mean absolute percentage error is 15%.

4.4.6.6.4. Confusion matrix

The Confusion Matrix component is a visualization tool typically used in supervised learning. This tool is used to calculate the classification accuracy of a confusion matrix model by comparing its results with measured values.

Procedure

1. Configure the confusion matrix parameters.

   The default settings are typically used. You can also select a target column and a prediction probability column. A prediction probability column is the target column generated by the Prediction component.

2. Connect the Confusion Matrix component and the Prediction component.

   > **Note** The parent node of the Confusion Matrix component must be a Prediction component. You can perform confusion matrix analysis only when a classification model is used.

3. Right-click the Confusion Matrix component and choose **View Evaluation Report**.

PAI command

```
PAI -name confusionmatrix
-project algo_public
-DoutputTableName="pai_temp_2954_24178_1"
-DlabelColName="age"
-DpredictionColName="prediction_result"
-DinputTableName="pai_temp_2954_24176_1";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.

Parameter	Description
outputTableName	The name of the output table.
labelColName	The name of the label column selected from the input table.
predictionColName	The name of the prediction result column.
inputTableName	The name of the input table for predicting results.

4.4.6.6.5. Multiclass classification evaluation

You can evaluate a multiclass classification model based on its predicted and actual results. The indicators include accuracy, kappa, and F1-Score.

Component description

The Multiclass Classification Evaluation component must be connected to a Prediction component and does not support regression models.

PAI command

```
PAI -name MultiClassEvaluation -project algo_public
    -DinputTableName="test_input"
    -DoutputTableName="test_output"
    -DlabelColName="label"
    -DpredictionColName="prediction_result"
    -Dlifecycle=30;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training.	-	All partitions of the input table are selected by default.
labelColName	Required. The name of the original label column in the input table. It must be a numerical value.	-	-
predictionColName	Required. The name of the label column in the prediction result table. It must be a numerical value.	-	-

Parameter	Description	Valid values	Default value
outputTableName	Required. The name of the output table.	-	-
predictionColName	Optional. The name of the probability column that lists prediction results. It must be in the {"A":0.2,"B":0.3} format.	-	-
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

4.4.6.7. Prediction

The Prediction component is used to make model-based predictions. The component has two inputs (training model and prediction data) and one output (prediction result). Conventional data mining algorithms often use this component for prediction.

Procedure

1. Connect all components.
2. Configure column settings.

Parameters

Parameter	Description
Feature Columns	The feature columns used for prediction. All feature columns are selected by default.
Reserved Columns	The columns reserved and exported to the prediction result.
Output Result Column	The default value is used.
Output Score Column	The default value is used.
Output Detail Column	The default value is used.

> **Note** Feature columns must be selected if data is in sparse format such as `k1:v1,k2:v2`.

3. After you configure the preceding parameters, right-click the Prediction component and choose **View Data** from the shortcut menu.

   The following three columns are appended to the prediction data:

   - predict_result: the prediction result column.
   - predict_score: the probability score in prediction results. This column is only appended

onto the outputs of binary classification models.

- predict_detail: the prediction result of each category. This column is only appended onto the outputs of binary classification models.

PAI command

```
PAI -name Prediction
-project algo_public
-DdetailColName="prediction_detail"
-DsplitCharacteristic="2"
-DappendColNames="age,campaign,pdays,previous,poutcome,emp_var_rate,cons_price_idx,cons_conf_
idx,euribor3m,nr_employed,y"
-DmodelName="xlab_m_random_forests_6036"
-DresultColName="prediction_result"
-DoutputTableName="pai_temp_675_6048_1"
-DscoreColName="prediction_score"
-DinputTableName="bank_data";
```

Parameters

Parameter	Description
name	The name of the component.
project	The name of the project. This parameter is used to specify the workspace of the algorithm. The default value is algo_public. If you change the name, the system reports an error.
detailColName	Optional. The name of the detail column in the output table. The default value is prediction_detail.
splitCharacteristic	Optional. The type of classification. The value *1* indicates binary classification. The value *2* indicates multiclass classification.
appendColNames	Optional. The names of columns in the input prediction table to be appended to the output table.
modelName	The name of the random forest model.
resultColName	Optional. The name of the result column in the output table. The default value is *prediction_result*.
outputTableName	The name of the output prediction table.
scoreColName	Optional. The name of the score column in the output table. The default value is *prediction_score*.
inputTableName	The name of the input prediction table.

4.4.7. Deep learning (must be activated separately)

4.4.7.1. Activate deep learning

The deep learning service is not a basic function of Apsara Stack Machine Learning Platform for AI. You must purchase it separately.

If you have already deployed the deep learning service, activate it by using the following procedure:

1. Log on to the Apsara Stack Machine Learning Platform for AI console.
2. Click **Settings** in the left-side navigation pane.
3. Click **General**. Under **Deep Learning**, select **Enable GPU Compute**.

4.4.7.2. Read OSS buckets

When using the **Read OSS Bucket** component on Machine Learning Platform for AI, you must assign the default system role **AliyunODPSPAIDefaultRole** to your DTplus service account. OSS buckets can be correctly read and written by algorithms of the machine learning platform only when the role is correctly assigned.

> **Note** The machine learning platform shares service accounts with MaxCompute, because it runs on the MaxCompute framework. During authorization, you must assign the default role to your MaxCompute service account.

You can use RAM authorization to grant OSS access permissions to Machine Learning Platform for AI. Click **Settings** to grant permission to read and write OSS data. For more information, see RAM authorization.

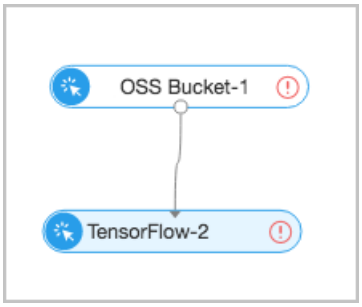
RAM authorization

1. Log on to the Machine Learning Platform For AI console, click **Settings** in the left-side navigation pane, and select **General**.
2. Under **OSS Authorization**, select **Authorize Machine Learning Platform for AI to access my OSS resources.**
3. The following page is displayed. Click **Click here to authorize access in RAM.** The RAM page is displayed.
4. Click **I Agree.**

Note To view details about the AliyunODPSPAIDefaultRole policy, log on to the RAM console. The default role AliyunODPSPAIDefaultRole contains the following permission information.

Permission (Action)	Description
oss:PutObject	Upload a file or folder object.
oss:GetObject	Obtain a file or folder object.
oss:ListObjects	Query file information.
oss:DeleteObjects	Delete an object.

5. Go back to the machine learning page and click **Refresh**. RAM information is automatically recorded to the components.
6. Use the deep learning framework. Connect the **Read OSS Bucket** component to the corresponding deep learning component to obtain permissions to read and write OSS data.



4.4.7.3. TensorFlow 1.4

TensorFlow (TF) is an open-source machine learning framework. It is easy to use for algorithm developers. The TF framework is integrated into Apsara Stack Machine Learning Platform for AI. You can write code and adjust computing resources flexibly by using the TF compute engine. The TF computing engine is a Graphics Processing Unit (GPU) cluster.

Parameters

- Parameter settings

Parameter settings

Parameter	Description
Python Code Files	The program execution files. Multiple files can be packaged and uploaded in the tar.gz format.
Primary Python File	Optional. The primary file in a compressed code file package.
Data Source Directory	The path of data sources. You can select Object Storage Service (OSS) data sources.

Parameter	Description
Configuration File Hyperparameters and Custom Parameters	Machine Learning Platform for AI Tensorflow allows you to use commands to pass in hyperparameter settings and try different learning rates and batch sizes during model testing.
Output Directory	The path of the output model.

- Tuning

  You can specify the number of GPUs based on the complexity of jobs.

PAI command

> **Note** You do not need to set all parameters. For the definitions of these parameters, see Parameters. We recommend that you do not directly copy the following command.

```
PAI -name tensorflow_ext140
-Dbuckets="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/"
-DgpuRequired="100"
-Darn="acs:ram::166408185518****:role/aliyunodpspaidefaultrole"
-Dscript="oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py";
```

The following table lists the descriptions of the parameters.

Parameters

Parameter	Description	Valid values	Default value
script	Required. The TF algorithm file. This file can be a single file or compressed as a tar.gz format package.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist_ext.py	N/A
entryFile	Optional. The name of the primary Python file. If the script is a compressed package in the tar.gz format, this parameter is required.	train.py	Null

Parameter	Description	Valid values	Default value
buckets	Required. The input OSS buckets. You can specify multiple buckets separated with commas (,). Each bucket must end with a forward slash (/).	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	Null
arn	Required. The Alibaba Cloud Resource Name (ARN) of an OSS object.	N/A	Null
gpuRequired	Required. This parameter indicates the number of GPUs to be used.	200	100
checkpointDir	Optional. The TF checkpoint directory.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/	Null
cluster	Optional.	A JSON format value. Quotation marks must be escaped.	Null
hyperParameters	Optional. The path of the command line hyperparameters.	oss://imagenet.oss-cn-shanghai-internal.aliyuncs.com/smoke_tensorflow/mnist/hyper_parameters.txt	Null

- script and entryFile are used to specify the TF algorithm script to be executed. If the algorithm is complex and divided into multiple files, you can package the files into a tar.gz file and use entryFile to specify the primary Python file.
- checkpointDir is used to specify the OSS path to be written by algorithms. You must specify the OSS path when you save TensorFlow models.
- buckets is used to specify the OSS path to be read by algorithms. To use OSS, you must specify arn.
- Distributed Machine Learning Platform for AI TensorFlow supports cluster. You can use cluster to specify the number of parameter servers and workers. cluster is in JSON format, and the quotation marks must be escaped. The JSON code must contain two keys: ps and worker. Both the ps and worker parameters contain count, gpu, cpu, and memory.

Keyword	Description	Default value	Remarks

Keyword	Description	Default value	Remarks
count	Required. The number of parameter servers or workers.	-	None
gpu	Optional. The number of GPUs allocated to each parameter server or worker. 100 represents a single GPU card.	For parameter servers, the default value is 0. For workers, the default value is 100.	If the number of GPUs allocated to each worker is set to 0, Machine Learning Platform for AI will reset the value to 100 to ensure the task is scheduled properly.
cpu	Optional. The number of CPUs allocated to each parameter server or worker. 100 represents a single CPU card.	600	None
memory	The memory size allocated to each parameter server or worker. 100 represents 100 MB.	30000	None

Examples

The MNIST digit classification set is a set of handwritten digits 1 through 9 that contains training and test sets for machine learning models.

1. Upload the Python execution files and training datasets to OSS. In this case, create a bucket on OSS in China (Shanghai) and name the bucket as tfmnist001. Upload the Python script and training data.
2. Drag and drop the **Read OSS Bucket** and **TensorFlow** components onto the canvas to create the following experiment. Set the region for the OSS bucket and configure RAM authorization.
3. Set the TensorFlow parameters. Set the paths for **Python Code Files**, **Primary Python File**, and **Data Source Directory**.
4. Click **Run** and wait for the experiment to complete running.
5. Right-click the **TensorFlow** component and view the running log.

4.4.8. Time series

4.4.8.1. x13_arima

Autoregressive Integrated Moving Average Model (ARIMA) is a well-known time series prediction method defined by Box and Jenkins in the early 1970s. This model is also called the Box-Jenkins model or the Box-Jenkins method. x13-arima is an ARIMA algorithm for seasonal adjustment based on the open-source X-13ARIMA-SEATS algorithm.

PAI command

```
pai -name x13_arima
    -project algo_public
    -DinputTableName=pai_ft_x13_arima_input
    -DseqColName=id
    -DvalueColName=number
    -Dorder=3,1,1
    -Dstart=1949.1
    -Dfrequency=12
    -Dseasonal=0,1,1
    -Dperiod=12
    -DpredictStep=12
    -DoutputPredictTableName=pai_ft_x13_arima_out_predict
    -DoutputDetailTableName=pai_ft_x13_arima_out_detail
```

Algorithm parameters

Parameters

Parameter	Description	Valid value	Default value
inputTableName	Required. The name of the input table.	Table name	N/A
inputTablePartitions	Optional. The partitions selected from the input table for training, in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
seqColName	Required. The name of the time series column.	Column name	This parameter is only used to sort the column specified by valueColName. The value does not affect the calculated results.

Parameter	Description	Valid value	Default value
valueColName	Required. The name of the value column.	Column name	N/A
groupColNames	Optional. The name of the stratification column. Separate multiple columns with commas (,), such as `col0,col1;` . A time series is created for each stratum.	Column name	N/A
order	Required. *p*, *d*, and *q* indicate the autoregressive coefficient, difference, and moving regression coefficient, respectively.	*p*, *d*, and *q* must be non-negative integers in the range of [0, 36].	N/A
start	Optional. The start date of a time series.	A string in the `year.seasonal` format, such as 1986.1 For more information, see the time series format section.	1.1
frequency	Optional. The frequency of a time series. Unit: months/year	A positive integer in the range of (0, 12) For more information, see the time series format section.	12
seasonal	Optional. *sp*, *sd*, and *sq* indicate the seasonal autoregressive coefficient, seasonal difference, and seasonal moving regression coefficient, respectively.	*sp*, *sd*, and *sq* must be non-negative integers in the range of [0, 36].	*seasonal* is not set by default.
period	Optional. The seasonal period.	A number in the range of (0, 100]	frequency
maxiter	Optional. The maximum number of iterations.	A positive integer	1500

Parameter	Description	Valid value	Default value
tol	Optional. The degree of tolerance.	A double type value	1e-5
predictStep	Optional. The number of prediction items.	A number in the range of (0, 365]	12
confidenceLevel	Optional. The prediction confidence level.	A number in the range of (0, 1)	0.95
outputPredictTableName	Required. The name of the output prediction table.	Table name	N/A
outputDetailTableName	Required. The name of the output detail table.	Table name	N/A
outputTablePartition	Optional. The partition in the output table.	Partition name	The output table is non-partitioned by default.
coreNum	Optional. The number of cores.	A positive integer used with memSizePerCore	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Time series format

- The start and frequency parameters specify the two time dimensions of data (valueColName): TS1 and TS2.
- The frequency parameter indicates the data frequency within a period, which equals the frequency of TS2 in each TS1.
- The start parameter must be in the `n1.n2` format. This indicates that the start date is the N2 TS2 in the N1 TS1.

Unit time	TS1	TS2	Frequency	Start date
12 months/year	Year	Month	12	1949.2 indicates the second month of year 1949.

Unit time	TS1	TS2	Frequency	Start date
Four quarters/year	Year	Quarter	4	1949.2 indicates the second quarter of year 1949.
Seven days/week	Day	Week	7	1949.2 indicates the second day of the1949th week.
1	Any time unit	1	1	1949.1 indicates the 1949th (year, day, or hour).

Example: value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15]

- `start=1949.3` and `frequency=12` indicate that the data frequency is monthly, and the prediction start date is 1950.06.

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949			1	2	3	4	5	6	7	8	9	10
1950	11	12	13	14	15							

- `start=1949.3` and `frequency=4` indicate that the data frequency is quarterly, and the prediction start date is 1953.02.

Year	Qtr1	Qtr2	Qtr3	Qtr4
1949			1	2
1950	3	4	5	6
1951	7	8	9	10
1952	11	12	13	14
1953	14			

- `start=1949.3` and `frequency=7` indicate that the data frequency is daily, and the prediction start date is 1951.04.

Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1949			1	2	3	4	5

Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1950	6	7	8	9	10	11	12
1951	13	14	15				

- `start=1949.1` and `frequency=1` indicate that the prediction start date is 1963.00 regardless of the time unit used.

Cycle	p1
1949	1
1950	2
1951	3
1951	4
1952	5
1953	6
1954	7
1955	8
1956	9
1957	10
1958	11
1959	12
1960	13
1961	14
1962	15

Examples

- Data used for testing: AirPassengers. The data set contains the number of passengers for international airlines each month from 1949 to 1960. It can be downloaded from https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/AirPassengers.html .

```
create table pai_ft_x13_arima_input(id bigint,number bigint);
tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;
```

- PAI command

```
pai -name x13_arima
  -project algo_public
  -DinputTableName=pai_ft_x13_arima_input
  -DseqColName=id
  -DvalueColName=number
  -Dorder=3,1,1
  -Dseasonal=0,1,1
  -Dstart=1949.1
  -Dfrequency=12
  -Dperiod=12
  -DpredictStep=12
  -DoutputPredictTableName=pai_ft_x13_arima_out_predict
  -DoutputDetailTableName=pai_ft_x13_arima_out_detail
```

- Output description
  - The columns of the output table specified by outputPredictTableName are as follows.

Column	Description
pdate	The prediction date.
forecast	The prediction result.
lower	The lower threshold of the prediction result when the confidence level is specified (default value: 0.95).
upper	The upper threshold of the prediction result when the confidence level is specified (default value: 0.95).

Output data

1	196101	444.445401291661	422.354385657496	466.536416925825
2	196102	420.971087699795	394.745551231805	447.196624167785
3	196103	453.432019696644	423.435661316528	483.428378076759
4	196104	490.922534668881	458.870488854835	522.974580482926
5	196105	503.877174753018	470.308964411549	537.445385094488
6	196106	566.536076521906	531.945171132091	601.126981911721
7	196107	652.606993945368	617.2596149799	687.954372910837
8	196108	639.841497141155	603.933582941945	675.749411340364
9	196109	542.341866147189	506.000630530659	578.683101763719
10	196110	494.745102803541	458.0614903363	531.428715270782
11	196111	426.635134341211	389.672783323704	463.597485358717
12	196112	468.722280768837	431.527372120416	505.917189417259

- The columns of the output table specified by outputDetailTableName are as follows.

Column	Description
key	"model" indicates the model. "evaluation" indicates the evaluation result. "parameters" indicates the training parameters. "log" indicates the training log.
summary	The storage details.

Output data

1	model	{ "comment": { "ma": "arima estimate", "mr": "regress...
2	evaluation	{ "comment": { "aic": "AIC", "aicc": "AICC (F-correcte...
3	paramters	{ "arima": { "d": 1, "isSeasonal": true, "p": 3, "period":...
4	log	1 Log for X-13ARIMA-SEATS program (Version 1.1...

- Model data (key=model)

operator	factor	period	lag	estimate	standard error
AR	Nonseasonal	1	1	0.6135	0.0928
AR	Nonseasonal	1	2	0.2403	0.1035
AR	Nonseasonal	1	3	-0.0732	0.0906
MA	Nonseasonal	1	1	0.9737	0.0376
MA	Seasonal	12	12	0.1051	0.1031

- Evaluation metrics (key=evaluation)

Name	Indicator
AIC	1019.6973
BIC	1036.9485
Hannan Quinn	1026.7072
Log likelihood	-503.8487
Effective number of observations	131
Number of observations	144
variance	127.0384

4.4.8.2. x13_auto_arima

ARIMA is described in x13_arima. The x13_auto_arima algorithm includes a process of automatic model selection.

The x13_auto_arima selection process is as follows:

- **Default model estimation**

  In the case of `frequency = 1`, the default model is `(0,1,1)`.

  In the case of `frequency > 1`, the default model is `(0,1,1)(0,1,1)`.

- **Identification of differencing orders**

  Skip this step if you have configured diff and SeasonalDiff.

  Use `Unit root test (wiki )` to determine the difference d and the seasonal difference D.

- **Identification of ARMA model orders**

  Select the optimal model based on BIC (wiki). The maxOrder and maxSeasonalOrder parameters are used in this step.

- **Comparison of identified model with default model**

  Use Ljung-Box Q statistic(wiki) to compare the models. If both models are unacceptable, use the `(3,d,1)(0,D,1)` model.

- **Final model checks**

PAI command

```
pai -name x13_auto_arima
    -project algo_public
    -DinputTableName=pai_ft_x13_arima_input
    -DseqColName=id
    -DvalueColName=number
    -Dstart=1949.1
    -Dfrequency=12
    -DpredictStep=12
    -DoutputPredictTableName=pai_ft_x13_arima_out_predict2
    -DoutputDetailTableName=pai_ft_x13_arima_out_detail2
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
seqColName	Required. The name of the time series column.	Column name	This parameter is only used to sort valueColNames. It is not relevant to the calculated output.
valueColName	Required. The name of the value column.	Column name	-
groupColNames	Optional. The name of the stratification column. Separate multiple columns with commas (,), such as `col0,col1;`. A time series is created for each group.	Column name	-
start	Optional. The start date of a time series.	A string in the format of `year.seasonal`, such as 1986.1 For more information, see the time series format section.	1.1
frequency	Optional. The frequency of a time series.	A positive integer in the range of (0, 12) For more information, see the time series format section.	The frequency is 12 months/year by default.

Parameter	Description	Valid values	Default value
maxOrder	Optional. The maximum values of p and q.	A positive integer in the range of [0, 4]	2
maxSeasonalOrder	Optional. The seasonal maximum values of p and q.	A positive integer in the range of [0, 2]	1
maxDiff	Optional. The maximum value of differential d.	A positive integer in the range of [0, 2]	2
maxSeasonalDiff	Optional. The maximum value of seasonal differential d.	A positive integer in the range of [0, 1]	1
diff	Optional. The differential d.	A positive integer in the range of [0, 2] If both diff and maxDiff are set, maxDiff is ignored. If diff is set, then seasonalDiff must also be set.	Default value: -1. This value indicates that diff is not specified by default.
seasonalDiff	Optional. The seasonal differential d.	A positive integer in the range of [0, 1] If both seasonalDiff and maxSeasonalDiff are set, maxSeasonalDiff is ignored.	Default value: -1. This value indicates that seasonalDiff is not specified by default.
maxiter	Optional. The maximum number of iterations.	A positive integer	1500
tol	Optional. The degree of tolerance.	A double type value	1e-5
predictStep	Optional. The number of prediction items.	A number in the range of (0, 365]	12
confidenceLevel	Optional. The prediction confidence level.	A number in the range of (0, 1)	0.95

Parameter	Description	Valid values	Default value
outputPredictTableName	Required. The name of the output prediction table.	Table name	-
outputDetailTableName	Required. The name of the output detail table.	Table name	-
outputTablePartition	Optional. The partitions in the output table.	Partition name	No partition is specified by default.
coreNum	Optional. The number of cores.	A positive integer used with memSizePerCore	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.

Time format

- The start and frequency parameters specify the two time dimensions of data (valueColName): TS1 and TS2.
- The frequency parameter indicates the data frequency within a period, which equals the frequency of TS2 in each TS1.
- The start parameter is in the format of `n1.n2`. This indicates that the start date is the N2 TS2 in the N1 TS1.

Unit time	ts1	ts2	Frequency	Start date
12	Year	Month	12	1949.2 indicates the second month of year 1949.
4	Year	Quarter	4	1949.2 indicates the second quarter of year 1949.
7	Day	Week	7	1949.2 indicates the second day of a week in year 1949.

Unit time	ts1	ts2	Frequency	Start date
1	Any time unit	1	1	1949.1 indicates the 1949th (year, day, or hour).

For example, value=[1,2,3,5,6,7,8,9,10,11,12,13,14,15].

- `start=1949.3` and `frequency=12` indicate that the data frequency is monthly, and the prediction start date is 1950.06.

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949			1	2	3	4	5	6	7	8	9	10
1950	11	12	13	14	15							

- `start=1949.3` and `frequency=4` indicate that the data frequency is quarterly, and the prediction start date is 1953.02.

Year	Qtr1	Qtr2	Qtr3	Qtr4
1949			1	2
1950	3	4	5	6
1951	7	8	9	10
1952	11	12	13	14
1953	14			

- `start=1949.3` and `frequency=7` indicate that the data frequency is daily, and the prediction start date is 1951.04.

Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1949			1	2	3	4	5
1950	6	7	8	9	10	11	12
1951	13	14	15				

- `start=1949.1` and `frequency=1` indicate that the end date is 1963.00.

Period	p1
1949	1

Period	p1
1950	2
1951	3
1951	4
1952	5
1953	6
1954	7
1955	8
1956	9
1957	10
1958	11
1959	12
1960	13
1961	14
1962	15

Examples

- Data used for testing: AirPassengers. This data set contains the number of passengers for international airlines each month from 1949 to 1960. It can be downloaded from https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/AirPassengers.html.

```
create table pai_ft_x13_arima_input(id bigint,number bigint);
tunnel upload data/airpassengers.csv pai_ft_x13_arima_input -h true;
```

- PAI command

```
pai -name x13_auto_arima
 -project algo_public
 -DinputTableName=pai_ft_x13_arima_input
 -DseqColName=id
 -DvalueColName=number
 -Dstart=1949.1
 -Dfrequency=12
 -DmaxOrder=4
 -DmaxSeasonalOrder=2
 -DmaxDiff=2
 -DmaxSeasonalDiff=1
 -DpredictStep=12
 -DoutputPredictTableName=pai_ft_x13_arima_auto_out_predict
 -DoutputDetailTableName=pai_ft_x13_arima_auto_out_detail
```

- Output description:
  - Output table: outputPredictTableName. The columns are as follows.

Column name	Description
pdate	The prediction date.
forecast	The prediction result.
lower	The lower threshold of the prediction result when the confidence level is confidenceLevel (default value: 0.95).
upper	The upper threshold of the prediction result when the confidence level is confidenceLevel (default value: 0.95).

Data:

	key	summary
1	model	{ "comment": { "ma": "arima estimate", "mr": "regress...
2	evaluation	{ "comment": { "aic": "AIC", "aicc": "AICC (F-correcte...
3	paramters	{ "arima": { "d": 1, "isSeasonal": true, "p": 3, "period":...
4	log	1 Log for X-13ARIMA-SEATS program (Version 1.1...

- Output table: outputDetailTableName. The columns are as follows.

Column name	Description
key	"model" indicates the model. "evaluation" indicates the evaluation result. "parameters" indicates the training parameters. "log" indicates the training log.
summary	Storage details.

4.4.9. Text analysis

4.4.9.1. Word splitting

Based on Alibaba Word Segmenter (AliWS), this component performs word splitting on documents specified by columns. Segmented words are separated with spaces. If you have set the part-of-speech (POS) tagging or semantic tagging parameters, the component outputs the word splitting results, POS tagging results, and semantic tagging results. Forward slashes `(/)` are used as delimiters for POS tagging. Vertical bars `(|)` are used as delimiters for semantic tagging. Only Chinese Taobao word segmentation and Internet word segmentation are supported.

Parameter settings

Word segmentation algorithms: CRF and UNIGRAM.

Parameters

Parameter	Description
Recognition Options	Specifies whether to recognize nouns with special meanings during word splitting.
Merge Options	Considers the terms used in certain industries as a whole without splitting.
Tokenizer	Allows you to select the Taobao word segmentation or Internet word segmentation. Taobao word segmentation is recommended.
Pos Tagger	Specifies whether to mark the part of speech for each word. If this parameter is specified, the part of speech for each word is marked in the output.

Examples

The following input table consists of the id column (document IDs) and the text column (document content).

PAI command

```
pai -name split_word
-project algo_public
-DinputTableName=doc_test
-DselectedColNames=content1,content2
-DoutputTableName=doc_test_split_word
-DinputTablePartitions="region=cctv_news"
-DoutputTablePartition="region=news"
-Dtokenizer=TAOBAO_CHN
-DenableDfa=true
-DenablePersonNameTagger=false
-DenableOrgnizationTagger=false
-DenablePosTagger=false
-DenableTelephoneRetrievalUnit=true
-DenableTimeRetrievalUnit=true
-DenableDateRetrievalUnit=true
-DenableNumberLetterRetrievalUnit=true
-DenableChnNumMerge=false
-DenableNumMerge=true
-DenableChnTimeMerge=false
-DenableChnDateMerge=false
-DenableSemanticTagger=true
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	The name of the input table.	-	-
selectedColNames	The names of the columns selected from the input table for word segmentation.	Separate multiple columns with commas (,).	-
outputTableName	The name of the output table.	-	-

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.
outputTablePartition	The partition in the output table.	-	The output table is non-partitioned by default.
tokenizer	The type of the classifier.	TAOBAO_CHN and INTERNET_CHN	Default value: TAOBAO_CHN. *TAOBAO_CHN* represents Taobao word segmentation. *INTERNET_CHN* represents Internet word segmentation.
enableDfa	Specifies whether to enable simple entity recognition.	true and false	true
enablePersonNameTagger	Specifies whether to enable personal name recognition.	true and false	false
enableOrgnizationTagger	Specifies whether to enable organization name recognition.	true and false	false
enablePosTagger	Specifies whether to enable part-of-speech tagging.	true and false	false
enableTelephoneRetrievalUnit	Specifies whether to enable retrieval unit configuration for telephone number recognition.	true and false	true

Parameter	Description	Valid values	Default value
enableTimeRetrievalUnit	Specifies whether to enable retrieval unit configuration for time ID recognition.	true and false	true
enableDateRetrievalUnit	Specifies whether to enable retrieval unit configuration for date ID recognition.	true and false	true
enableNumberLetterRetrievalUnit	Specifies whether to enable retrieval unit configuration for number and letter recognition.	true and false	true
enableChnNumMerge	Specifies whether to merge Chinese numbers into a retrieval unit.	true and false	false
enableNumMerge	Specifies whether to merge regular numbers into a retrieval unit.	true and false	true
enableChnTimeMerge	Specifies whether to merge Chinese time into a semantic unit.	true and false	false
enableChnDateMerge	Specifies whether to merge Chinese dates into a semantic unit.	true and false	false
enableSemanticTagger	Specifies whether to enable semantic tagging.	true and false	false

4.4.9.2. Deprecated word filtering

Deprecated word filtering is a preprocessing method in text analysis. This method is used to filter out the noise in word splitting results, such as of, yes, and ah.

Parameter settings

The left and right input ports are as follows:

- Input table, which is a word splitting result table for filtering. Parameter: inputTableName
- Deprecated word table, which is a one-column table with each row containing a deprecated word. Parameter: noiseTableName

PAI command

```
PAI -name FilterNoise
-project algo_public
-DinputTableName="test_input"
-DnoiseTableName="noise_input"
-DoutputTableName="test_output"
-DselectedColNames="words_seg1,words_seg2"
-Dlifecycle=30
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	-	All partitions in the input table are selected by default.
noiseTableName	Required. The name of the deprecated word table.	A one-column table with each row containing a deprecated word	-
noiseTablePartitions	Optional. The partitions selected from the deprecated word table.	-	All partitions in the table are selected by default.
outputTableName	Required. The name of the output table.	-	-
selectedColNames	Required. The name of the column to be filtered. Separate multiple columns with commas (,).	-	-
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of (0, 65536)	Automatically calculated.

4.4.9.3. String similarity

String similarity calculation is a basic operation in machine learning that is used in information retrieval, natural language processing, and bioinformatics. This algorithm supports five methods to calculate similarity: Levenshtein distance, longest common substring, string subsequence kernel, cosine, and simhash_hamming. It also supports two input methods: string-to-string calculation and top N calculation.

PAI command

```
PAI -name string_similarity
-project algo_public
-DinputTableName="pai_test_string_similarity"
-DoutputTableName="pai_test_string_similarity_output"
-DinputSelectedColName1="col0"
-DinputSelectedColName2="col1";
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
outputTableName	Required. The name of the output table.	-	-
inputSelectedColName1	Optional. The name of the first column for similarity calculation.	-	By default, the first string type column in the table is selected.
inputSelectedColName2	Optional. The name of the second column for similarity calculation.	-	The second string type column in the table is selected by default.
inputAppendColNames	Optional. The names of columns appended to the output table.	-	No column is appended by default.
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	-	The whole table is selected by default.

Parameter	Description	Valid values	Default value
outputColName	Optional. The name of the similarity column in the output table. The column name can be up to 128 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.	-	output
method	Optional. The similarity calculation method.	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, simhash_hamming_sim, minhash_sim, and hash_jaccard_sim	levenshtein_sim
lambda	Optional. The weight of the matching string. This parameter takes effect when similarityType is set to ssk.	(0, 1)	0.5
k	Optional. The length of the substring. This parameter takes effect when similarityType is set to ssk or cosine.	(0, 100)	2
kVec	Optional. The number of MinHash instances.	A positive integer	2
b	Optional. The number of buckets.	A positive integer	1
seed	Optional. The random seed used in a MinHash instance.	A positive integer	0
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of (0, 65536)	Automatically calculated.

Examples

- SQL statement to generate data:

```
create table pai_ft_string_similarity_input
as select * from
(select 0 as id, "Beijing" as col0,
"Beijing" as col1 from dual union all
select 1 as id,
"Beijing" as col0,
"Beijing Shanghai" as col1 from dual union all
select 2 as id,
"Beijing" as col0,
"Beijing Shanghai Hongkong" as col1 from dual )tmp;
```

- PAI command

```
PAI -name string_similarity
-project sre_mpi_algo_dev
-DinputTableName=pai_ft_string_similarity_input
-DoutputTableName=pai_ft_string_similarity_output
-DinputSelectedColName1=col0
-DinputSelectedColName2=col1
-Dmethod=simhash_hamming
-DinputAppendColNames=col0,col1;
```

- Output description
  - Output obtained by using the simhash_hamming method:

col0	col1	output
beijing	beijing	0
beijing	beijing shanghai	6
beijing	beijing shanghai xianggang	13

  - Output obtained by using the simhash_hamming_sim method:

col0	col1	output
beijing	beijing	1
beijing	beijing shanghai	0.90625
beijing	beijing shanghai xianggang	0.796875

4.4.9.4. Convert row, column, and value to KV pair

This component converts rows, columns, and values into KV pairs. A row, column, and value set is defined as XXD or XXL, where X can represent any type, D represents Double, and L represents Bigint. The row, column, and value set is converted into KV format (row, [col_id:value]). The row and value types are consistent with the original input data. The col_id type is Bigint, and the column is mapped to col_id based on the index table.

PAI command

```
PAI -name triple_to_kv
-project algo_public
-DinputTableName=test_data
-DoutputTableName=test_kv_out
-DindexOutputTableName=test_index_out
-DidColName=id
-DkeyColName=word
-DvalueColName=count
-DinputTablePartitions=ds=test1
-DindexInputTableName=test_index_input
-DindexInputKeyColName=word
-DindexInputKeyIdColName=word_id
-DkvDelimiter=:
-DpairDelimiter=;
-Dlifecycle=3
```

Parameters

Parameters

Parameter	Description	Default value
inputTableName	Required. The name of the input table.	The input table cannot be empty.
idColName	Required. The name of the column to be retained after the table is converted into a KV table.	-
keyColName	Required. The name of the key column in the KV table.	-
valueColName	Required. The name of the value column in the KV table.	-
outputTableName	Required. The name of the output KV table.	-

Parameter	Description	Default value
indexOutputTableName	Required. The name of the index table for the output keys.	-
indexInputTableName	Optional. The name of the input index table.	No index table is set by default. The table cannot be empty and it does not need to contain indexes for all of the output keys.
indexInputKeyColName	Optional. The name of the key column in the input index table.	No key column is specified by default. This parameter is required if indexInputTableName is set.
indexInputKeyIdColName	Optional. The name of the index column in the input index table.	No index column is specified by default. This parameter is required if indexInputTableName is set.
inputTablePartitions	Optional. The partitions in the input table.	No partition is specified by default. Only one partition can be input.
kvDelimiter	Optional. The delimiter used to separate the key and value.	The default delimiter is a colon (:).
pairDelimiter	Optional. The delimiter used to separate KV pairs.	The default delimiter is a semicolon (;).
lifecycle	Optional. The lifecycle of the output table.	No lifecycle is set by default.
coreNum	Optional. The number of cores.	-1
memSizePerCore	Optional. The memory size of each core. Valid values: 100 to 65536.	-1

Examples

- SQL statement to generate data:

```
drop table if exists triple2kv_test_input;
create table triple2kv_test_input as
select       *     from
(      select '01' as id, 'a' as word,
10 as count from dual        union all
select '01' as id, 'b' as word,
20 as count from dual        union all
select '01' as id, 'c' as word,
30 as count from dual        union all
select '02' as id,
'a' as word,
100 as count from dual        union all
select '02' as id, 'd' as word,
200 as count from dual        union all
select '02' as id, 'e' as word,
300 as count from dual     ) tmp;
```

- PAI command

```
PAI -name triple_to_kv
-project algo_public
-DinputTableName=triple2kv_test_input
-DoutputTableName=triple2kv_test_input_out
-DindexOutputTableName=triple2kv_test_input_index_out
-DidColName=id
-DkeyColName=word
-DvalueColName=count
-Dlifecycle=1;
```

Input description

Input table

Input description

id	word	count
01	a	10
01	b	20
01	c	30

Output description

- The output KV table is as follows, where custom KV delimiters can be used.

Output description

id	key_value
01	1:10;2:20;3:30

- The output index table that contains indexes for the words is as follows.

Output index table

key	key_id
a	1
b	2
c	3

4.4.9.5. String similarity - Top N

String similarity calculation is a basic operation in machine learning that is used in information retrieval, natural language processing, and bioinformatics. This algorithm supports five methods to calculate similarity: Levenshtein distance, longest common substring, string subsequence kernel, cosine, and simhash_hamming. It also supports two input methods: string-to-string calculation and top N calculation.

PAI command

```
PAI -name string_similarity_topn
-project algo_public
-DinputTableName="pai_test_string_similarity_topn"
-DoutputTableName="pai_test_string_similarity_topn_output"
-DmapTableName="pai_test_string_similarity_map_topn"
-DinputSelectedColName="col0"
-DmapSelectedColName="col1";
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
mapTableName	Required. The name of the mapping table.	-	-
outputTableName	Required. The name of the output table.	-	-

Parameter	Description	Valid values	Default value
inputSelectedColName	Optional. The name of the column selected from the left table for similarity calculation.	-	The first string type column in the table is selected by default.
mapSelectedColName	Optional. The name of the column selected from the mapping table for similarity calculation. The similarities between each row in the left table and all strings in the mapping table are calculated, and the top N entries are output.	-	The first string type column in the table is selected by default.
inputAppendColNames	Optional. The names of columns appended to the output table from the input table.	-	No column is appended by default.
inputAppendRenameColNames	Optional. The aliases of columns appended to the output table from the input table. This parameter takes effect when inputAppendColNames is specified.	-	No alias is specified by default.
mapAppendColNames	Optional. The names of columns appended to the output table from the mapping table.	-	No column is appended by default.
mapAppendRenameColNames	Optional. The aliases of columns appended to the output table from the mapping table.	-	No alias is specified by default.
inputTablePartitions	Optional. The partitions selected from the input table.	-	The whole table is selected by default.
mapTablePartitions	Optional. The partitions in the mapping table.	-	The whole table is selected by default.

Parameter	Description	Valid values	Default value
outputColName	Optional. The name of the similarity column in the output table. The column name can be up to 128 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.	-	output
method	Optional. The similarity calculation method.	levenshtein_sim, lcs_sim, ssk, cosine, simhash_hamming_sim, minhash_sim, and hash_jaccard_sim	levenshtein_sim
lambda	Optional. The weight of the matching string. This parameter takes effect when similarityType is set to ssk.	(0, 1)	0.5
k	Optional. The length of the substring. This parameter takes effect when similarityType is set to ssk or cosine.	(0, 100)	2
kVec	Optional. The number of MinHash instances.	A positive integer	2
b	Optional. The number of buckets.	A positive integer	1
seed	Optional. The random seed used in a MinHash instance.	A positive integer	0
topN	Optional. The number of similarity maximums to be output.	(0, +∞)	10
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.

Parameter	Description	Valid values	Default value
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of (0, 65536)	Automatically calculated.

Examples

- SQL statement to generate data:

```
create table pai_ft_string_similarity_topn_input
as select * from
(select 0 as id,
"Beijing" as col0 from dual union all
select 1 as id,
"Beijing Shanghai" as col0 from dual union all
select 2 as id,
"Beijing Shanghai Hongkong" as col0 from dual )tmp;
```

- PAI command

```
PAI -name string_similarity_topn
-project sre_mpi_algo_dev
-DinputTableName=pai_ft_string_similarity_topn_input
-DmapTableName=pai_ft_string_similarity_topn_input
-DoutputTableName=pai_ft_string_similarity_topn_output
-DinputSelectedColName=col0
-DmapSelectedColName=col0
-DinputAppendColNames=col0
-DinputAppendRenameColNames=input_col0
-DmapAppendColNames=col0
-DmapAppendRenameColNames=map_col0
-Dmethod=simhash_hamming_sim;
```

- Output.

input_col0	map_col0	output
beijing	beijing	1
beiji beijing	beijing shanghai	0.90625
beijing	beijing shanghai xianggang	0.796875
beijing shanghai	beijing shanghai	1
beijing shanghai	beijing	0.90625
beijing shanghai	beijing shanghai xianggang	0.828125
beijing shanghai xianggang	beijing shanghai xianggang	1
beijing shanghai xianggang	beijing shanghai	0.828125
beijing shanghai xianggang	beijing	0.796875

4.4.9.6. N-gram counting

N-gram counting is a step in language model training. N-grams are generated based on words.The number of the corresponding N-grams in all corpora is counted. The N-gram counting model counts the number of N-grams in all documents rather than in a single document. For more information, see ngram-count.

PAI command

```
PAI -name ngram_count
    -project algo_public
    -DinputTableName=pai_ngram_input
    -DoutputTableName=pai_ngram_output
    -DinputSelectedColNames=col0
    -DweightColName=weight
    -DcoreNum=2
    -DmemSizePerCore=1000;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	N/A
outputTableName	Required. The name of the output table.	Table name	N/A
inputSelectedColNames	Optional. The names of columns selected from the input table.	Column name	The first character type column is selected by default.
weightColName	Optional. The name of the weight column.	Column name	1

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table.	Partition name	The whole table is selected by default.
countTableName	Optional. The name of the former N-gram counting output table. This table is merged into the output result.	Table name	N/A
countWordColName	Optional. The name of the word column in the counting table.	Column name	The second column is selected by default.
countCountColName	Optional. The name of the counting column in the counting table.	Column name	The third column is selected by default.
countTablePartitions	Optional. The partitions in the counting table.	Partition name	N/A
vocabTableName	Optional. The name of the bag-of-words table. The words that are not contained in the bag-of-words table are marked with \<unk\>.	Table name	N/A
vocabSelectedColName	Optional. The name of the bag-of-words column.	Column name	The first character type column is selected by default.
vocabTablePartitions	Optional. The partitions in the bag-of-words table.	Partition name	N/A
order	Optional. The maximum length of N-grams.	N/A	3
lifecycle	Optional. The lifecycle of the output table.	A positive integer	N/A
coreNum	Optional. The number of cores.	A positive integer	N/A
memSizePerCore	Optional. The memory size of each core.	A positive integer	N/A

4.4.9.7. Text summarization

Automatic summarization uses computers to automatically extract summaries from a source document. A summary is a simple, concise, and short document that completely and accurately describes the content of a certain document. This TextRank-based algorithm generates summaries by extracting existing sentences in the document.

PAI command

```
PAI -name TextSummarization
-project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-DdocIdCol="doc_id"
-DsentenceCol="sentence"
-DtopN=2
-Dlifecycle=30;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	-	All partitions in the input table are selected by default.
outputTableName	Required. The name of the output table.	-	-
docIdCol	Required. The name of the document ID column.	-	-
sentenceCol	Required. The sentence column.	Only one column can be specified.	-
topN	Optional. The top N key sentences to be output.	-	3
similarityType	Optional. The method used to calculate sentence similarity.	lcs_sim, levenshtein_sim, cosine, and ssk	lcs_sim

Parameter	Description	Valid values	Default value
lambda	Optional. The weight of the matching string. This parameter takes effect when similarityType is set to ssk.	(0, 1)	0.5
k	Optional. The length of the substring. This parameter takes effect when similarityType is set to ssk or cosine.	(0, 100)	2
dampingFactor	Optional. The damping factor.	(0, 1)	0.85
maxIter	Optional. The maximum number of iterations.	[1, +]	100
epsilon	Optional. The convergence coefficient.	(0, ∞)	0.000001
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	A positive integer	Automatically calculated.

The sentence similarity options are as follows:

- lcs_sim: The formula is `1.0 - (Length of the longest common subsequence)/max(len(A), len(B))`.
- levenshtein_sim: The formula is `1.0 - (Levenshtein distance)/max(len(A), len(B))`.
- cosine: See `Lodhi, Huma; Saunders, Craig; Shawe-Taylor, John; Cristianini, Nello; Watkins, Chris (2002). "Text classification using string kernels". Journal of Machine Learning Research: 419-444`.
- ssk: See `Leslie, C.; Eskin, E.; Noble, W.S. (2002), The spectrum kernel: A string kernel for SVM protein classification 7, pp. 566-575`.

Note `A` and `B` indicate two strings, and `len(A)` indicates the length of string A.

Output format description

The output table contains the doc_id and abstract columns, as shown in Output table example.

Output table example

doc_id	abstract
1000894	In 2008, the Shanghai Stock Exchange published disclosure guidelines for the corporate social responsibility of listed companies. Three types of companies were urged to disclose their CSR reports, and other qualified listed companies were encouraged to voluntarily disclose their CSR reports. In 2012, a total of 379 listed companies making up a 40% of all listed companies disclosed CSR reports. Of those companies, 305 were mandated to disclose CSR reports and and 75 voluntarily disclosed CSR reports. According to Hu Ruyin, Shanghai Stock Exchange will explore how to expand the scope of CSR report disclosure, revise and refine the guidelines on disclosure of the CSR reports, and encourage more organizations to promote CSR product innovation.

4.4.9.8. Keyword extraction

Keyword extraction is one of the important technologies in natural language processing. It is used to extract keywords from a document. This algorithm is based on TextRank, a variation of the PageRank algorithm used to describe the relationship between webpages. This algorithm uses the relationship between certain words to construct a network, calculate the importance of each word, and determine words with larger weights as keywords.

PAI command

```
PAI -name KeywordsExtraction
-DinputTableName=maple_test_keywords_basic_input
-DdocIdCol=docid -DdocContent=word
-DoututTableName=maple_test_keywords_basic_output
-DtopN=19;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-

Parameter	Description	Valid values	Default value
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions are selected by default.
outputTableName	Required. The name of the output table.	-	-
docIdCol	Required. The name of the document ID column.	Only one column can be specified.	-
docContent	Required. The word column.	Only one column can be specified.	-
topN	Optional. The number of top N keywords to be output. If this number is smaller than the number of keywords, all keywords are output.	-	5
windowSize	Optional. The window size of the TextRank algorithm.	-	2
dumpingFactor	Optional. The damping factor of the TextRank algorithm.	-	0.85
maxIter	Optional. The maximum number of iterations of the TextRank algorithm.	-	100
epsilon	Optional. The convergence residual threshold of the TextRank algorithm.	-	0.000001

Parameter	Description	Valid values	Default value
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

Examples

The words in the input table are separated with spaces, and deprecated words and all punctuations are filtered out.

Examples

docid: string	word: string
doc0	The blended-wing-body aircraft is a new direction for the future development in the aviation field Many research institutions inside and outside China have carried out research on the blended-wing-body aircraft while its fully automated shape optimization algorithm has become a new hot topic Based on the existing research achievements inside and outside China common modeling and flow solver tools have been analyzed and compared The geometric modeling grid flow field solver and shape optimization modules have been designed The pros and cons between different algorithms have been compared to achieve the optimized shape of the blended-wing-body aircraft in the conceptual design stage Geometric modeling and grid generation module are achieved based on the transfinite interpolation algorithm and spline based grid generation method The flow solver module includes the finite difference solver the finite element solver and the panel method solver The finite difference solver includes mathematical modeling of the potential flow the derivation of the Cartesian grid based variable step length difference scheme Cartesian grid generation and indexing algorithm the Cartesian grid based Neumann boundary conditions expression form derivation are achieved based on finite element difference solver The aerodynamic parameters of a two-dimensional

docid: string	word: string
	airfoil are calculated based on the finite difference solver The finite element solver includes potential flow modeling based on the variational principle of the finite element theory the derivation of the two-dimensional finite element Kutta conditional least squares based speed solving algorithm Gmsh based two-dimensional field grid generator of airfoil with wakes design The aerodynamic parameters of a two-dimensional airfoil are calculated based on the finite element solver The panel method solver includes modeling and automatic wake generation the design of the three-dimensional flow solver of the blended-wing-body drag estimation based on the Blasius solution solver implemented in the Fortran language a mixed compilation of Python and Fortran OpenMP and CUDA based acceleration algorithm The aerodynamic parameters of a three-dimensional wing body are calculated based on the panel method solver The shape optimization module includes free form deformation algorithm genetic algorithms differential evolution algorithm Aircraft surface area calculation algorithm is based on the moments integration algorithm The volume of an aircraft calculation algorithm is based on VKT data visualization format tool

PAI command

```
PAI -name KeywordsExtraction
-DinputTableName=maple_test_keywords_basic_input
-DdocIdCol=docid -DdocContent=word
-DoututTableName=maple_test_keywords_basic_output
-DtopN=19;
```

Input/output description

Output table description

docid	keywords	weight
doc0	Based on	0.041306752223538405
doc0	Algorithm	0.03089845626854151
doc0	Modeling	0.021782865850562882
doc0	Grid	0.020669749212693957
doc0	Solver	0.020245609506360847

docid	keywords	weight
doc0	Aircraft	0.019850761705313365
doc0	Research	0.014193732541852615
doc0	Finite element	0.013831122054200538
doc0	Solving	0.012924593244133104
doc0	Module	0.01280216562287212
doc0	Derivation	0.011907588923852495
doc0	Shape	0.011505456605632607
doc0	Difference	0.011477831662367547
doc0	Flow	0.010969269350293957
doc0	Design	0.010830986516637251
doc0	Implementation	0.010747536556701583
doc0	Two-dimensional	0.010695570768457084
doc0	Development	0.010527342662670088
doc0	New	0.010096978306668461

4.4.9.9. Sentence splitting

You can split sentences in a document by punctuation. This component is used to preprocess text summarizations. It splits text such that each row contains only a single sentence.

PAI command

```
PAI -name SplitSentences
-project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-DdocIdCol="doc_id"
-DdocContent="content"
-Dlifecycle=30
```

Parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	-	All partitions in the input table are selected by default.
outputTableName	Required. The name of the output table.	-	-
docIdCol	Required. The name of the document ID column.	-	-
docContent	Required. The name of the document content column.	Only one column can be specified.	-
delimiter	Optional. A set of characters used to determine the end of a sentence.	-	The default delimiter set contains the period (.), question mark (!), and exclamation mark (?).
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	A positive integer	Automatically calculated.

Output format description

The output table contains the doc_id and sentence columns, as shown in Output table example.

Output table example

doc_id	sentence
1000894	In 2008, the Shanghai Stock Exchange published disclosure guidelines for the corporate social responsibility of listed companies. Three types of companies were urged to disclose their CSR reports, and other qualified listed companies were encouraged to voluntarily disclose their CSR reports.

doc_id	sentence
1000894	In 2012, a total of 379 listed companies making up a 40% of all listed companies disclosed CSR reports. Of those companies, 305 were mandated to disclose CSR reports and and 75 voluntarily disclosed CSR reports.

4.4.9.10. Semantic vector distance

You can calculate the extension words or sentences of the specified words or sentences based on the calculated semantic vectors, such as word vectors calculated by the Word2Vec component. The extension words or sentences are a set of vectors closest to a certain vector. The following example shows how to generate a list of words that are most similar to the word that you entered based on the word vectors calculated by the Word2Vec component.

PAI command

```
PAI -name SemanticVectorDistance
-project algo_public
-DinputTableName="test_input"
-DoutputTableName="test_output"
-DidColName="word"
-DvectorColNames="f0,f1,f2,f3,f4,f5"
-Dlifecycle=30
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for calculation.	-	All partitions in the input table are selected by default.
outputTableName	Required. The name of the output table.	-	-
idTableName	Optional. The name of the vector ID table for vector calculation. The table contains only one column and each row stores a vector ID.	-	No vector ID table is specified by default. This means that all vectors in the input table are calculated.

Parameter	Description	Valid values	Default value
idTablePartitions	Optional. The partitions selected from the ID table for calculation.	-	All partitions are selected by default.
idColName	Required. The name of the ID column.	-	3
vectorColNames	Optional. A list of vector column names, such as f1, f2,...	-	-
topN	Optional. The number of the closest vectors to output.	[1, +∞]	5
distanceType	Optional. The distance calculation method.	euclidean, cosine, and manhattan	euclidean
distanceThreshold	Optional. The distance threshold. Only the distances between two vectors that do not exceed this threshold are output.	(0, +∞)	∞
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core.	A positive integer	Automatically calculated.

Examples

The output table contains the original_id, near_id, distance, and rank columns.

original_id	near_id	distance	rank
hello	hi	0.2	1
hello	xxx	xx	2
Man	Woman	0.3	1
Man	xx	xx	2
..	...	...	...

4.4.9.11. Document similarity

This algorithm calculates the similarity between two text documents by comparing the similarities of documents or sentences separated by spaces. This algorithms functions similar to how the similarity of strings is calculated.

PAI command

```
PAI -name doc_similarity
-project algo_public
-DinputTableName="pai_test_doc_similarity"
-DoutputTableName="pai_test_doc_similarity_output"
-DinputSelectedColName1="col0"
-DinputSelectedColName2="col1"
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	-	-
outputTableName	Required. The name of the output table.	-	-
inputSelectedColName1	Optional. The name of the first column for similarity calculation.	-	By default, the first string type column in the table is selected.
inputSelectedColName2	Optional. The name of the second column for similarity calculation.	-	The name of the second string type column in the table is selected by default.
inputAppendColNames	Optional. The names of columns appended to the output table.	-	No column is appended by default.
inputTablePartitions	Optional. The partitions selected from the input table.	-	The whole table is selected by default.

Parameter	Description	Valid values	Default value
outputColName	Optional. The name of the similarity column in the output table. The column name can be up to 128 characters in length and can contain letters, digits, and underscores (_). It must start with a letter.	-	output
method	Optional. The similarity calculation method.	levenshtein, levenshtein_sim, lcs, lcs_sim, ssk, cosine, simhash_hamming, and simhash_hamming_sim	levenshtein_sim
lambda	Optional. The weight of the matching word pair. This parameter takes effect if similarityType is set to ssk.	(0, 1)	0.5
k	Optional. The length of the substring. This parameter takes effect if similarityType is set to ssk or cosine.	(0, 100)	2
kVec	Optional. The number of MinHash instances.	A positive integer	2
b	Optional. The number of buckets.	A positive integer	1
seed	Optional. The random seed used in a MinHash instance.	A positive integer	0
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size of each core. Unit: MB.	A positive integer in the range of (0, 65536)	Automatically calculated.

Examples

- SQL statement to generate data:

```
drop table if exists pai_doc_similarity_input;
create table pai_doc_similarity_input as
select * from (
select 0 as id,
"Beijing and Shanghai" as col0,
"Beijing and Shanghai" as col1 from dual union all
select 1 as id,
"Beijing and Shanghai" as col0,
"Beijing, Shanghai, and Hong Kong" as col1 from dual )tmp;
```

- PAI command

```
drop table if exists pai_doc_similarity_output;
PAI -name doc_similarity
-project algo_public
-DinputTableName=pai_doc_similarity_input
-DoutputTableName=pai_doc_similarity_output
-DinputSelectedColName1=col0
-DinputSelectedColName2=col1
-Dmethod=levenshtein_sim
-DinputAppendColNames=id,col0,col1;
```

- Input description: pai_doc_similarity_input

ID	col0	col1
1	Beijing and Shanghai	Beijing, Shanghai, and Hong Kong
0	Beijing and Shanghai	Beijing and Shanghai

- Output description: pai_doc_similarity_output

ID	col0	col1	Output
1	Beijing and Shanghai	Beijing, Shanghai, and Hong Kong	0.6666666666666667
0	Beijing and Shanghai	Beijing and Shanghai	1.0

4.4.9.12. PMI

Mutual information (MI) is a measure of information in the information theory. It can be regarded as the amount of information contained in a random variable about another variable, or the reduction in uncertainty of a random variable due to the known random variable.

This algorithm is used to count the co-occurrence of all words in several documents and calculate the point mutual information（PMI）. PMI definition: `PMI(x,y)=ln(p(x,y)/(p(x)p(y)))=ln(#(x,y)D/(#x#y))`.

- `#(x,y)` indicates the number of pair(x,y).
- D indicates the total number of pairs.
- If x and y appear in the same window, the output is #x+=1;#y+=1;#(x,y)+=1.

PAI command

```
PAI -name PointwiseMutualInformation
    -project algo_public
    -DinputTableName=maple_test_pmi_basic_input
    -DdocColName=doc
    -DoutputTableName=maple_test_pmi_basic_output
    -DminCount=0
    -DwindowSize=2
    -DcoreNum=1
    -DmemSizePerCore=110;
```

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
outputTableName	Required. The name of the output table.	Table name	-
docColName	Required. The name of the document column after word splitting, where words are separated with spaces.	Column name	-
windowSize	Optional. The window size. For example, the value 5 refers to the five words adjacent on the right of the current word. Words that appear in the window are considered related to the current word.	[1, sentence length]	The whole row is selected by default.

Parameter	Description	Valid values	Default value
minCount	The minimum word truncation frequency. Words that appear for a number of times less than this value are filtered out.	[0, 2e63]	5
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	Partition name	All partitions are selected by default.
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set by default.
coreNum	Optional. The number of cores.	This parameter is used with memSizePerCore. The value must be a positive integer in the range of [1, 9999].	Automatically calculated.
memSizePerCore	The memory size of each core. Unit: MB.	A positive integer in the range of [1024, 65536]	Automatically calculated.

Examples

- **Data generation**

doc:string
w1 w2 w3 w4 w5 w6 w7 w8 w8 w9
w1 w3 w5 w6 w9
w0
w0 w0
w9 w1 w9 w1 w9

- **PAI command**

```
PAI -name PointwiseMutualInformation
    -project algo_public
    -DinputTableName=maple_test_pmi_basic_input
    -DdocColName=doc
    -DoutputTableName=maple_test_pmi_basic_output
    -DminCount=0
    -DwindowSize=2
    -DcoreNum=1
    -DmemSizePerCore=110;
```

- **Output description**

Output table

word1	word2	word1_count	word2_count	co_occurrences_count	pmi
w0	w0	2	2	1	2.0794415416798357
w1	w1	10	10	1	-1.1394342831883648
w1	w2	10	3	1	0.06453852113757116
w1	w3	10	7	2	-0.08961215868968704
w1	w5	10	8	1	-0.916290731874155
w1	w9	10	12	4	0.06453852113757116
w2	w3	3	7	1	0.4212134650763035
w2	w4	3	4	1	0.9808292530117262
w3	w4	7	4	1	0.13353139262452257
w3	w5	7	8	2	0.13353139262452257

word1	word2	word1_count	word2_count	co_occurrences_count	pmi
w3	w6	7	7	1	-0.42608439531090014
w4	w5	4	8	1	0
w4	w6	4	7	1	0.13353139262452257
w5	w6	8	7	2	0.13353139262452257
w5	w7	8	4	1	0
w5	w9	8	12	1	-1.0986122886681098
w6	w7	7	4	1	0.13353139262452257
w6	w8	7	7	1	-0.42608439531090014
w6	w9	7	12	1	-0.9650808960435872
w7	w8	4	7	2	0.8266785731844679
w8	w8	7	7	1	-0.42608439531090014
w8	w9	7	12	2	-0.2719337154836418
w9	w9	12	12	2	-0.8109302162163288

4.4.9.13. Word frequency statistics

Based on the word splitting results, this component outputs the words in their original order and calculates the frequency that a word occurs in the document (docContent) specified by the document ID column (docId).

Parameter settings

Input parameters: docId column and docContent column generated by the Word Splitting component.

Two output parameters:

- Output port 1: The output table contains the id, word, and count columns.

  count: indicates the frequency that a word occurs in each document.

- Output port 2: The output table contains the id and word columns.

  The table output by the second output port lists words in order of occurrence in the document. The table does not calculate the frequency of the occurrence. Therefore, a word may have multiple table entries in the same document. The output table format is compatible with the Word2Vec component.

Examples

In the Alibaba Cloud word splitting data, the two columns in the output table are used as the input parameters for word frequency calculation.

- Select the docId column: id.
- Select the docContent column: After the word frequency calculation is performed, the result is displayed by output port 1 on this component.

PAI command

```
pai -name doc_word_stat
-project algo_public
-DinputTableName=doc_test_split_word
-DdocId=id
-DdocContent=content
-DoutputTableNameMulti=doc_test_stat_multi
-DoutputTableNameTriple=doc_test_stat_triple
-DinputTablePartitions="region=cctv_news"
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	The name of the input table.	-	-
docId	The name of the document ID column.	Only one column can be specified.	-
docContent	The name of the document content column.	Only one column can be specified.	-

Parameter	Description	Valid values	Default value
outputTableNameMulti	The name of the output table that lists words in the document content after word splitting. Documents are specified by the docId column and their contents are specified by the docContent column. The words are listed in the order that they occur within the documents.	-	-
outputTableNameTriple	The name of the output table that lists the words and the frequency of the occurrence of these words in the documents. The documents are specified by the docId column and their contents are specified by the docContent column.	-	-
inputTablePartitions	Optional. The partitions selected from the input table for training, in the format of `partition_name=value`. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	-	All partitions in the input table are selected by default.

4.4.9.14. TF-IDF

Term frequency-inverse document frequency (TF-IDF) is typically used as a weighting technology in information retrieval and text mining. TF-IDF is a statistical method to evaluate the importance of a word for a document in a collection or corpus. The importance of a word increases as the frequency that it occurs within the document increases. The importance decreases as the frequency that the word occurs in the corpus increases. TF-IDF is frequently used by search engines as a tool in scoring and ranking the correlation between documents and user queries.

- For more information, see TF-IDF in Wikipedia.
- The TF-IDF component is used to calculate the TF-IDF value of each word that appears in a collection of documents based on word frequency statistics.

Examples

The output table in the example of the word frequency statistics component is used as the input table for the TF-IDF component. The corresponding parameter settings are as follows:

- Select the document ID column: id
- Select the word column: word
- Select the word count column: count

The output table contains the following columns: docid, word, word_count (frequency that a certain word occurs in the current document), total_word_count (total number of words in the current document), doc_count (total number of documents that contain the current word), total_doc_count (total number of documents), tf, idf.

PAI command

```
pai -name tfidf
-project algo_public
-DinputTableName=rgdoc_split_triple_out
-DdocIdCol=id
-DwordCol=word
-DcountCol=count
-DoutputTableName=rg_tfidf_out;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartitions	Optional. The partitions selected from the input table for word splitting.	This value must be in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	All partitions in the input table are selected by default.
docIdCol	Required. The name of the document ID column.	Only one column can be specified.	-
wordCol	Required. The name of the word column.	Only one column can be specified.	-
countCol	Required. The name of the count column.	Only one column can be specified.	-
outputTableName	Required. The name of the output table.	Table name	-

Parameter	Description	Valid values	Default value
outputTablePartition	The partitions in the output table.	Partition name	The output table is non-partitioned by default.

4.4.9.15. PLDA

Latent Dirichlet Allocation (LDA) is a topic model that outputs the topic of a document. It outputs the topic of each document based on probability distribution. LDA is an unsupervised learning algorithm that does not require a manually tagged training set. Instead, it only requires a set of documents and the number of topics (k). It is used in text mining, including text topic recognition, text classification, and text similarity calculation.

Parameter settings

Parameters

Parameter	Description
Topics	The number of topics output by LDA.
Alpha	The AlphaPrior Dirichlet distribution parameter of `P(z/d)` .
Beta	The AlphaPrior Dirichlet distribution parameter of `P(w/z)` .
Burn-in Iterations	The number of burn-in iterations. The parameter value must be less than the total number of iterations. The default value is 100.
Total Iterations	Optional. The total number of iterations. The parameter value must be a positive integer. The default value is 150.

Note z represents topics, w represents words, and d represents documents.

Input and output settings

- Input:

  The data must be in the sparse matrix format. For more information about the format, see the data format description section. You can use the Convert Row, Column, and Value to KV Pair component to convert the data. Input format as shows the following picture.

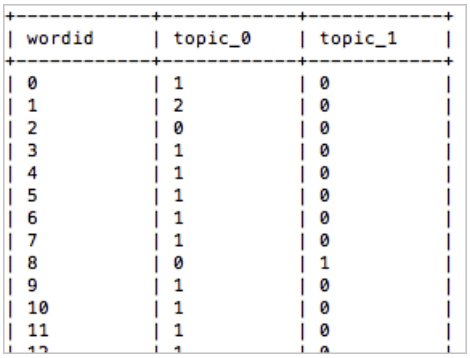
```
| id          | features |
+-------------+----------+
| 2           | 38:3.0,39:1.0,40:3.0,41:1.0,42:1.0,43:2.0,44:1.0,45:1.0,46:1.0,47:1.0,48:1.0,49:2.0,50:1.0,51:1.0,52:1.0,53:1.0,54:1.0,55:1.0,5
6:1.0,57:1.0,58:1.0,59:1.0,60:1.0,61:1.0,62:1.0,63:1.0,64:1.0,65:1.0,66:1.0,67:1.0,68:1.0,69:1.0,70:1.0,71:1.0,72:1.0,73:1.0,74:1.0,75:1.0,76:
1.0,77:2.0 |
| 1           | 0:1.0,1:2.0,3:1.0,4:1.0,5:1.0,6:1.0,7:1.0,8:1.0,9:1.0,10:1.0,11:1.0,12:1.0,13:1.0,14:2.0,15:1.0,16:1.0,17:1.0,18:1.0,19:1.0,20:
1.0,21:1.0,22:1.0,23:1.0,24:1.0,25:1.0,26:1.0,27:1.0,28:1.0,29:1.0,30:1.0,31:1.0,32:1.0,33:1.0,34:1.0,35:1.0,36:2.0,39:2.0,77:3.0 |
+-------------+----------+
```

  - Column 1: the ID of a document.
  - Column 2: KV data of the word and how frequently it occurs.

- Output:

The following tables are generated in sequence: topic-word frequency contribution table, P(w/z) table, P(z/w) table, P(d/z) table, P(z/d) table, and P(z) table.

The following picture shows the output format of the topic-word frequency contribution table.



wordid	topic_0	topic_1
0	1	0
1	2	0
2	0	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	0	1
9	1	0
10	1	0
11	1	0

PAI command

```
pai -name PLDA
-project algo_public
-DinputTableName=lda_input
-DtopicNum=10
-topicWordTableName=lda_output;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartitions	Optional. The partitions selected from the input table for word splitting.	This value must be in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	All partitions in the input table are selected by default.
selectedColNames	Optional. The names of the columns selected from the input table for LDA.	Separate multiple columns with commas (,).	All columns in the input table are selected by default.

Parameter	Description	Valid values	Default value
topicNum	Required. The number of topics.	[2, 500]	-
kvDelimiter	Optional. The delimiter used to separate the key and value.	Space, comma (,), and colon (:)	The default delimiter is a colon (:).
itemDelimiter	Optional. The delimiter used to separate keys.	Space, comma (,), and colon (:)	The default delimiter is a space.
alpha	Optional. The prior Dirichlet distribution parameter of P(z/d).	(0, ∞)	0.1
beta	Optional. The prior Dirichlet distribution parameter of P(w/z).	(0, ∞)	0.01
topicWordTableName	Required. The name of the topic-word frequency contribution table.	Table name	-
pwzTableName	Optional. The name of the P(w/z) table.	Table name	No P(w/z) table is output by default.
pzwTableName	Optional. The name of the P(z/w) table.	Table name	No P(z/w) table is output by default.
pdzTableName	Optional. The name of the P(d/z) table.	Table name	No P(d/z) table is output by default.
pzdTableName	Optional. The name of the P(z/d) table.	Table name	No P(z/d) table is output by default.
pzTableName	Optional. The name of the P(z) table.	Table name	No P(z) table is output by default.
burnInIterations	Optional. The number of burn-in iterations.	A positive integer	This value must be smaller than the total number of iterations. The default value is 100.
totalIterations	Optional. The number of iterations.	A positive integer	150

4.4.9.16. Word2Vec

Word2Vec is an open-source algorithm used to convert words into vectors. By training neural networks, Word2Vec can map words to K-dimensional space vectors and map word vectors to semantics.

For information about the Google Word2Vec toolkit, visit https://code.google.com/p/word2vec/.

Parameter settings

- Dimension of Word Features: We recommend a value from 0 to 1000.
- Downsampling Threshold: We recommend a value from 1e-3 to 1e-5.
- Input: inputs a word column and a vocabulary.
- Output: generates a word vector table and a vocabulary.

PAI command

```
pai -name Word2Vec
-project algo_public
-DinputTableName=w2v_input
-DwordColName=word
-DoutputTableName=w2v_output;
```

Algorithm parameters

Parameters

Parameter	Description	Valid values	Default value
inputTableName	Required. The name of the input table.	Table name	-
inputTablePartitions	Optional. The partitions selected from the input table for word splitting.	The parameter value must be in the `partition_name=value` format. To specify multiple partitions, use the following format: `name1=value1/name2=value2`. Separate multiple partitions with commas (,).	All partitions in the input table are selected by default.
wordColName	Required. The name of the word column. Each row in the word column contains a single word. `</s>` is used to break lines in the corpus.	Column name	-

Parameter	Description	Valid values	Default value
inVocabularyTableName	Optional. The name of the input word list, which contains the wordcount output of inputTableName.	Table name	Word count is performed for the input table by default.
inVocabularyPartitions	Optional. The partitions in the input word list.	Partition name	By default, all partitions in the table specified by inVocabularyTableName are selected.
layerSize	Optional. The dimension of word features.	0 to 1000	100
cbow	Optional. The language model.	1: cbow. 0: skip-gram.	0
window	Optional. The size of the word window.	A positive integer	5
minCount	Optional. The minimum frequency of word truncation.	A positive integer	5
hs	Optional. This parameter specifies whether to use hierarchical softmax.	1: Hierarchical softmax is used. 0: Hierarchical softmax is not used.	1
negative	Optional. The negative sampling.	0: Negative sampling is unavailable. Recommended value range: 5 to 10.	0
sample	Optional. The downward sampling threshold.	0 or smaller values: downward sampling is unavailable. Recommended value range: 1e-3 to 1e-5.	0
alpha	Optional. The initial learning rate.	A value greater than 0	0.025
iterTrain	Optional. The number of training iterations.	A value greater than or equal to 1	1

Parameter	Description	Valid values	Default value
randomWindow	Optional. This parameter specifies whether to randomly set the size of the window.	1: The window size is randomly generated.The window size value will range from 1 to 5. 0: The window size is determined by the window parameter.	1
outVocabularyTableName	Optional. The name of the output word list.	Table name	No Output Word List is generated by default.
outVocabularyPartition	Optional. The partition in the output word list.	Partition name	The output word list is non-partitioned by default.
outputTableName	Required. The name of the output table.	Table name	-
outputPartition	Optional. The information about partitions in the output table.	Partition name	The output table is non-partitioned by default.

4.4.10. Network analysis

4.4.10.1. K-core

The k-core of a graph is the largest subgraph in which every vertex is connected to at least k other vertices within the subgraph. The coreness of a vertex is k if it belongs to the k-core but is not included in the (k+1)-core. Therefore, the coreness of a vertex whose degree is 1 must be 0. The graph coreness is equal to that of the vertex with the largest coreness.

Parameter settings

k: Required. The value of the coreness. Default value: 3.

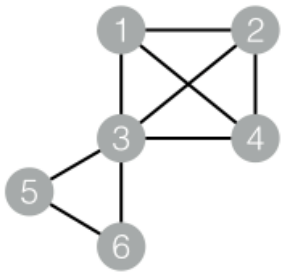
Examples - Testing data

SQL statement to generate data:

```
drop table if exists KCore_func_test_edge;
create table KCore_func_test_edge as
select * from (
select '1' as flow_out_id,
'2' as flow_in_id from dual   union all
select '1' as flow_out_id,
'3' as flow_in_id from dual   union all
select '1' as flow_out_id,
'4' as flow_in_id from dual   union all
select '2' as flow_out_id,
'3' as flow_in_id from dual   union all
select '2' as flow_out_id,
'4' as flow_in_id from dual   union all
select '3' as flow_out_id,
'4' as flow_in_id from dual   union all
select '3' as flow_out_id,
'5' as flow_in_id from dual   union all
select '3' as flow_out_id,
'6' as flow_in_id from dual   union all
select '5' as flow_out_id,
'6' as flow_in_id from dual )tmp;
```

Graph structure shows the group structure.

Graph structure



Set K to 2. Output shows the output.

Output

```
+-------+-------+
| node1 | node2 |
+-------+-------+
| 1     | 2     |
| 1     | 3     |
| 1     | 4     |
| 2     | 1     |
| 2     | 3     |
| 2     | 4     |
| 3     | 1     |
| 3     | 2     |
| 3     | 4     |
| 4     | 1     |
| 4     | 2     |
| 4     | 3     |
+-------+-------+
```

PAI command

```
pai -name KCore
-project algo_public
-DinputEdgeTableName=KCore_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=KCore_func_test_result
-Dk=2;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the edge table.	Yes	-
toVertexCol	The end vertex column in the edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-

Parameter	Description	Required	Default value
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64
k	The number of cores.	Yes	3

4.4.10.2. Single-source shortest path

The single-source shortest path (SSSP) refers to the shortest path between a vertex and all other vertices as calculated by the Dijkstra algorithm.

Parameter settings

Start Vertex ID: Required. The ID of the start vertex used to calculate the shortest paths.

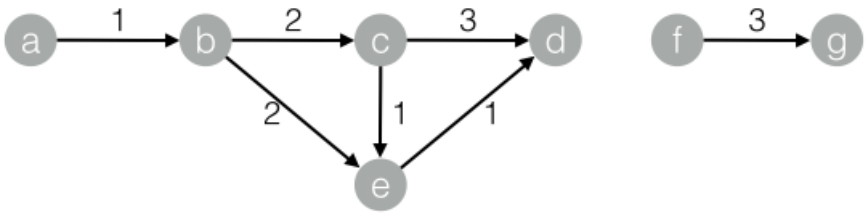
Examples - Testing data

SQL statement to generate data:

```
drop table if exists SSSP_func_test_edge;
create table SSSP_func_test_edge
as select
flow_out_id,flow_in_id,edge_weight from (
select "a" as flow_out_id,
"b" as flow_in_id,
1.0 as edge_weight from dual    union all
select "b" as flow_out_id,
"c" as flow_in_id,
2.0 as edge_weight from dual    union all
select "c" as flow_out_id,
"d" as flow_in_id,
1.0 as edge_weight from dual    union all
select "b" as flow_out_id,
"e" as flow_in_id,
2.0 as edge_weight from dual    union all
select "e" as flow_out_id,
"d" as flow_in_id,
1.0 as edge_weight from dual    union all
select "c" as flow_out_id,
"e" as flow_in_id,
1.0 as edge_weight from dual    union all
select "f" as flow_out_id,
"g" as flow_in_id,
3.0 as edge_weight from dual    union all
select "a" as flow_out_id,
"d" as flow_in_id,
4.0 as edge_weight from dual ) tmp ;
```

Graph structure shows the graph structure.

Graph structure



Output

```
+------------+------------+------------+--------------+
| start_node | dest_node  | distance   | distance_cnt |
+------------+------------+------------+--------------+
| a          | b          | 1.0        | 1            |
| a          | c          | 3.0        | 1            |
| a          | d          | 4.0        | 3            |
| a          | a          | 0.0        | 0            |
| a          | e          | 3.0        | 1            |
+------------+------------+------------+--------------+
```

PAI command

```
pai -name SSSP
-project algo_public
-DinputEdgeTableName=SSSP_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=SSSP_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=edge_weight
-DstartVertex=a;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-

Parameter	Description	Required	Default value
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64
startVertex	The ID of the start vertex.	Yes	-
hasEdgeWeight	Specifies whether the edges in the input edge table have weights.	No	false
edgeWeightCol	The edge weight column in the input edge table.	No	-

4.4.10.3. PageRank

The PageRank algorithm is used to sort and calculate the rankings of web pages based on their link sources.

Features

The basic principle of the PageRank algorithm is as follows: The more web pages that direct to a web page, the more importance or higher quality the web page has. In addition to the number of links directing to a web page, the weight of the web page and the number of outgoing links are also considered during page ranking. For a social network of users, the edge weight is an important factor in addition to the influence of the users. For example, a Sina Weibo user is more likely to have influence on their family, friends, classmates, and colleagues than they will on followers with a weaker relationship. In the social network, the edge weight is equivalent to the user-to-user relationship strength index. The PageRank formula with connection weight is as follows:

$$W(A) = (1 - d) + d * (\sum_{i} W(i) * C(Ai))$$

In the formula, W(i) represents the weight of node i, C(A,i) represents the link weight, and d represents the damping coefficient. W is the influence index of each user and represents the node weight after the algorithm iteration becomes stable.

Parameter settings

Maximum Iterations: Optional. The number of iterations performed before the algorithm automatically converges. Default value: 30.

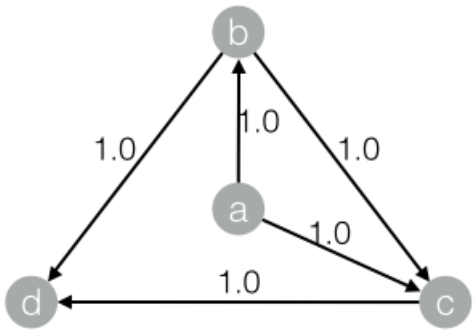
Examples - Testing data

SQL statement to generate data:

```
drop table if exists PageRankWithWeight_func_test_edge;
create table PageRankWithWeight_func_test_edge
as select * from (
select 'a' as flow_out_id,
'b' as flow_in_id,
1.0 as weight from dual    union all
select 'a' as flow_out_id,
'c' as flow_in_id,
1.0 as weight from dual    union all
select 'b' as flow_out_id,
'c' as flow_in_id,
1.0 as weight from dual    union all
select 'b' as flow_out_id,
'd' as flow_in_id,
1.0 as weight from dual    union all
select 'c' as flow_out_id,
'd' as flow_in_id,1.0 as weight from dual )tmp ;
```

Graph structure shows the graph structure.

Graph structure



Output

node	weight
a	0.0375
b	0.06938
c	0.12834
d	0.20556

PAI command

```
pai -name PageRankWithWeight
-project algo_public
-DinputEdgeTableName=PageRankWithWeight_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=PageRankWithWeight_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=weight
-DmaxIter 100;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

Parameter	Description	Required	Default value
hasEdgeWeight	Specifies whether the edges in the input edge table have weights.	No	false
edgeWeightCol	The edge weight column in the input edge table.	No	-
maxIter	The maximum number of iterations.	No	30

4.4.10.4. Label propagation clustering

Graph clustering is used to divide a graph into subgraphs based on the topology of the graph so that the links between the nodes in a subgraph are more than the links between the subgraphs. The label propagation algorithm (LPA) is a graph-based semi-supervised machine learning algorithm. The labels of a node (community) depend on those of the neighboring nodes. The degree of dependence is determined by the similarity between nodes. Data becomes stable by iterative propagation update.

Parameters

Maximum Iterations: Optional. The maximum number of iterations. Default value: 30.

Examples - Testing data

SQL statement to generate data:

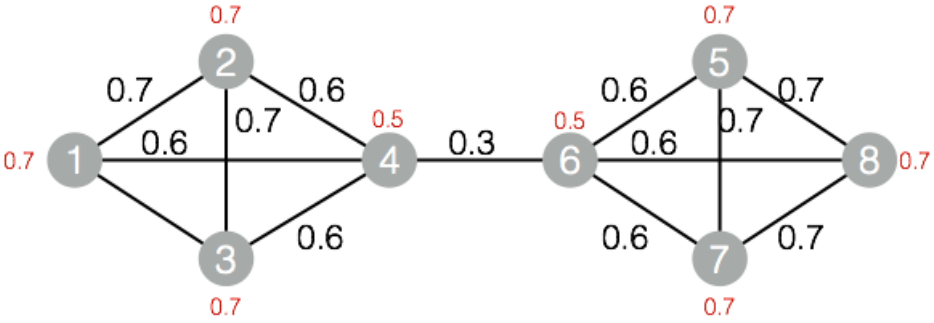
```
drop table if exists LabelPropagationClustering_func_test_edge;
create table LabelPropagationClustering_func_test_edge
as select * from (
select '1' as flow_out_id,
'2' as flow_in_id,
0.7 as edge_weight from dual    union all
select '1' as flow_out_id,
'3' as flow_in_id,
0.7 as edge_weight from dual    union all
select '1' as flow_out_id,
'4' as flow_in_id,
0.6 as edge_weight from dual    union all
select '2' as flow_out_id,
'3' as flow_in_id,
0.7 as edge_weight from dual    union all
select '2' as flow_out_id,
'4' as flow_in_id,
```

```
0.6 as edge_weight from dual    union all
select '3' as flow_out_id,
'4' as flow_in_id,
0.6 as edge_weight from dual    union all
select '4' as flow_out_id,
'6' as flow_in_id,
0.3 as edge_weight from dual    union all
select '5' as flow_out_id,
'6' as flow_in_id,
0.6 as edge_weight from dual    union all
select '5' as flow_out_id,
'7' as flow_in_id,
0.7 as edge_weight from dual    union all
select '5' as flow_out_id,
'8' as flow_in_id,
0.7 as edge_weight from dual    union all
select '6' as flow_out_id,
'7' as flow_in_id,
0.6 as edge_weight from dual    union all
select '6' as flow_out_id,
'8' as flow_in_id,
0.6 as edge_weight from dual    union all
select '7' as flow_out_id,
'8' as flow_in_id,
0.7 as edge_weight from dual )tmp ;
drop table if exists LabelPropagationClustering_func_test_node;
create table LabelPropagationClustering_func_test_node
as select * from (
select '1' as node,
0.7 as node_weight from dual    union all
select '2' as node,
0.7 as node_weight from dual    union all
select '3' as node,
0.7 as node_weight from dual    union all
select '4' as node,
0.5 as node_weight from dual    union all
select '5' as node,
0.7 as node_weight from dual    union all
select '6' as node,
0.5 as node_weight from dual    union all
select '7' as node,
```

```
0.7 as node_weight from dual     union all
select '8' as node,
0.7 as node_weight from dual )tmp ;
```

Group structure shows the group structure.

Group structure



Output

```
+-------+-------------+
| node | group_id    |
+-------+-------------+
| 1    | 1           |
| 2    | 1           |
| 3    | 1           |
| 4    | 1           |
| 5    | 5           |
| 6    | 5           |
| 7    | 5           |
| 8    | 5           |
+-------+-------------+
```

PAI command

```
pai -name LabelPropagationClustering
-project algo_public
-DinputEdgeTableName=LabelPropagationClustering_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DinputVertexTableName=LabelPropagationClustering_func_test_node
-DvertexCol=node
-DoutputTableName=LabelPropagationClustering_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=edge_weight
-DhasVertexWeight=true
-DvertexWeightCol=node_weight
-DrandSelect=true
-DmaxIter=100;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
inputVertexTableName	The name of the input vertex table.	Yes	-
inputVertexTablePartitions	The partitions in the input vertex table.	No	The whole table is selected by default.
vertexCol	The vertex column in the input vertex table.	Yes	-
outputTableName	The name of the output table.	Yes	-

Parameter	Description	Required	Default value
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64
hasEdgeWeight	Specifies whether the edges in the input edge table have weights.	No	false
edgeWeightCol	The edge weight column in the input edge table.	No	-
hasVertexWeight	Specifies whether the vertices in the input vertex table have weights.	No	false
vertexWeightCol	The vertex weight column in the input vertex table.	No	-
randSelect	Specifies whether the maximum label value is to be randomly selected.	No	false
maxIter	The maximum number of iterations.	No	30

4.4.10.5. Label propagation classification

Label propagation classification is a semi-supervised classification algorithm. It uses the label information of labeled nodes to predict the label information for unlabeled nodes.

Features

During algorithm execution, the labels of each node are propagated to the neighboring nodes based on the similarity between the nodes. In each step of propagation, a node updates its labels based on the labels of the neighboring nodes so that the node is more similar to the neighboring nodes. The higher the similarity, the more labeling influences the neighboring nodes have on that node, and the easier it is for the labels to be propagated. During label propagation, the labels of the labeled data remain unchanged. These labels serve as sources for propagation to the unlabeled data.

After the iterations end, the probability distributions of similar nodes tend to be similar. These nodes can be classified into the same category. This completes the label propagation.

Parameter settings

Damping factor: The default value is 0.8. Convergence factor: The default value is 0.000001.

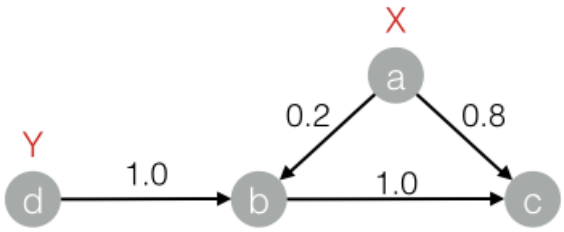
Examples - Testing data

SQL statement to generate data:

```
drop table if exists LabelPropagationClassification_func_test_edge;
create table LabelPropagationClassification_func_test_edge
as select * from (
select 'a' as flow_out_id,
'b' as flow_in_id,
0.2 as edge_weight from dual    union all
select 'a' as flow_out_id,
'c' as flow_in_id,
0.8 as edge_weight from dual    union all
select 'b' as flow_out_id,
'c' as flow_in_id,
1.0 as edge_weight from dual    union all
select 'd' as flow_out_id,
'b' as flow_in_id,
1.0 as edge_weight from dual )tmp ;
drop table if exists LabelPropagationClassification_func_test_node;
create table LabelPropagationClassification_func_test_node
as select * from (
select 'a' as node,
'X' as label,
1.0 as label_weight from dual    union all
select 'd' as node,
'Y' as label,
1.0 as label_weight from dual )tmp ;
```

Graph structure shows the graph structure.

Graph structure



Output

node	tag	weight
a	X	1.0
b	X	0.16667
b	Y	0.83333
c	X	0.53704
c	Y	0.46296
d	Y	1.0

PAI command

```
pai -name LabelPropagationClassification
-project algo_public
-DinputEdgeTableName=LabelPropagationClassification_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DinputVertexTableName=LabelPropagationClassification_func_test_node
-DvertexCol=node
-DvertexLabelCol=label
-DoutputTableName=LabelPropagationClassification_func_test_result
-DhasEdgeWeight=true
-DedgeWeightCol=edge_weight
-DhasVertexWeight=true
-DvertexWeightCol=label_weight
-Dalpha=0.8
-Depsilon=0.000001;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-

Parameter	Description	Required	Default value
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
inputVertexTableName	The name of the input vertex table.	Yes	-
inputVertexTablePartitions	The partitions in the input vertex table.	No	The whole table is selected by default.
vertexCol	The vertex column in the input vertex table.	Yes	-
vertexLabelCol	The vertex label column in the input vertex table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64
hasEdgeWeight	Specifies whether the edges in the input edge table have weights.	No	false
edgeWeightCol	The edge weight column in the input edge table.	No	-

Parameter	Description	Required	Default value
hasVertexWeight	Specifies whether the vertices in the input vertex table have weights.	No	false
vertexWeightCol	The vertex weight column in the input vertex table.	No	-
alpha	The damping coefficient.	No	0.8
epsilon	The convergence coefficient.	No	0.000001
maxIter	The maximum number of iterations.	No	30

4.4.10.6. Modularity

Modularity is used to measure the structure of the community network. It measures the closeness of the communities divided from a network structure. A value larger than 0.3 represents an obvious community structure.

Examples - Testing data

For more information, see Label propagation clustering.

Output

```
+--------------+
| val          |
+--------------+
| 0.4230769    |
+--------------+
```

PAI command

```
pai -name Modularity
-project algo_public
-DinputEdgeTableName=Modularity_func_test_edge
-DfromVertexCol=flow_out_id
-DfromGroupCol=group_out_id
-DtoVertexCol=flow_in_id
-DtoGroupCol=group_in_id
-DoutputTableName=Modularity_func_test_result;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
fromGroupCol	The start vertex group in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
toGroupCol	The end vertex group in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.10.7. Maximum connected subgraph

In an undirected graph G, vertex A is connected to vertex B if a path exists between the two vertices. Graph G contains several subgraphs. Each vertex is connected to other vertices in the same subgraph. Vertices in different subgraphs are not connected. In this case, the subgraphs of graph G are called maximum connected subgraphs.

Features

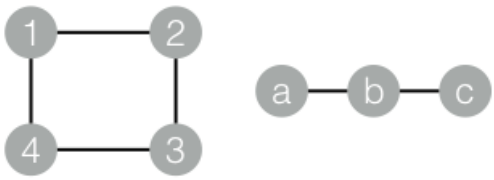
Examples - Testing data

SQL statement to generate data:

```
drop table if exists MaximalConnectedComponent_func_test_edge;
create table MaximalConnectedComponent_func_test_edge
as select * from (
select '1' as flow_out_id,
'2' as flow_in_id from dual   union all
select '2' as flow_out_id,
'3' as flow_in_id from dual   union all
select '3' as flow_out_id,
'4' as flow_in_id from dual   union all
select '1' as flow_out_id,
'4' as flow_in_id from dual   union all
select 'a' as flow_out_id,
'b' as flow_in_id from dual   union all
select 'b' as flow_out_id,
'c' as flow_in_id from dual )tmp;
drop table if exists MaximalConnectedComponent_func_test_result;
create table MaximalConnectedComponent_func_test_result (   node string,   grp_id string );
```

Graph structure shows the graph structure.

Graph structure



Output

node	grp_id
1	4
2	4
3	4
4	4
a	c
b	c
c	c

PAI command

```
pai -name MaximalConnectedComponent
-project algo_public
-DinputEdgeTableName=MaximalConnectedComponent_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=MaximalConnectedComponent_func_test_result;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.10.8. Vertex clustering coefficient

This coefficient is used to calculate the peripheral density of a vertex in an undirected graph G. The density of a star network is 0, and that of a fully meshed network is 1.

Parameter settings

maxEdgeCnt: Optional. If the node degree is larger than the value of this parameter, sampling is required. Default value: 500.

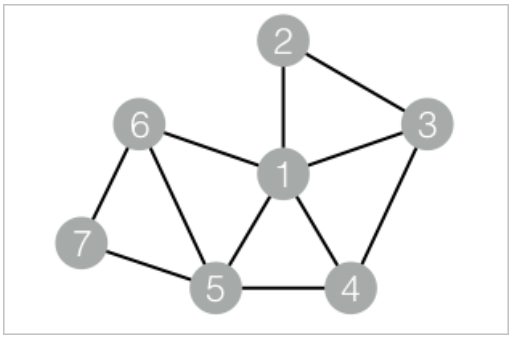
Examples - Testing data

SQL statement to generate data:

```
drop table if exists NodeDensity_func_test_edge;
create table NodeDensity_func_test_edge
as select * from (
select '1' as flow_out_id,
'2' as flow_in_id from dual   union all
select '1' as flow_out_id,
'3' as flow_in_id from dual   union all
select '1' as flow_out_id,
'4' as flow_in_id from dual   union all
select '1' as flow_out_id,
'5' as flow_in_id from dual   union all
select '1' as flow_out_id,
'6' as flow_in_id from dual   union all
select '2' as flow_out_id,
'3' as flow_in_id from dual   union all
select '3' as flow_out_id,
'4' as flow_in_id from dual   union all
select '4' as flow_out_id,
'5' as flow_in_id from dual   union all
select '5' as flow_out_id,
'6' as flow_in_id from dual   union all
select '5' as flow_out_id,
'7' as flow_in_id from dual   union all
select '6' as flow_out_id,
'7' as flow_in_id from dual )tmp;
drop table if exists NodeDensity_func_test_result;
create table NodeDensity_func_test_result (   node string,   node_cnt bigint,   edge_cnt bigint,   density
double,   log_density double );
```

Graph structure shows the graph structure.

Graph structure



Output

```
1,5,4,0.4,1.45657
2,2,1,1.0,1.24696
3,3,2,0.66667,1.35204
4,3,2,0.66667,1.35204
5,4,3,0.5,1.41189
6,3,2,0.66667,1.35204
7,2,1,1.0,1.24696
```

PAI command

```
pai -name NodeDensity
-project algo_public
-DinputEdgeTableName=NodeDensity_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=NodeDensity_func_test_result
-DmaxEdgeCnt=500;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-

Parameter	Description	Required	Default value
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
maxEdgeCnt	If the node degree is larger than the value of this parameter, sampling is required.	No	500
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.10.9. Edge clustering coefficient

This coefficient is used to calculate the peripheral density of each edge in an undirected graph G.

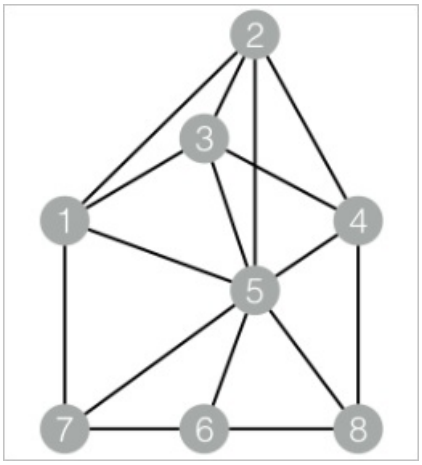
Examples - Testing data

SQL statement to generate data:

```
drop table if exists EdgeDensity_func_test_edge;
create table EdgeDensity_func_test_edge
as select * from (
select '1' as flow_out_id,
'2' as flow_in_id from dual   union all
select '1' as flow_out_id,
'3' as flow_in_id from dual   union all
select '1' as flow_out_id,
'5' as flow_in_id from dual   union all
select '1' as flow_out_id,
'7' as flow_in_id from dual   union all
select '2' as flow_out_id,
'5' as flow_in_id from dual   union all
select '2' as flow_out_id,
'4' as flow_in_id from dual   union all
select '2' as flow_out_id,
'3' as flow_in_id from dual   union all
select '3' as flow_out_id,
'5' as flow_in_id from dual   union all
select '3' as flow_out_id,
'4' as flow_in_id from dual   union all
select '4' as flow_out_id,
'5' as flow_in_id from dual   union all
select '4' as flow_out_id,
'8' as flow_in_id from dual   union all
select '5' as flow_out_id,
'6' as flow_in_id from dual   union all
select '5' as flow_out_id,
'7' as flow_in_id from dual   union all
select '5' as flow_out_id,
'8' as flow_in_id from dual   union all
select '7' as flow_out_id,
'6' as flow_in_id from dual   union all
select '6' as flow_out_id,
'8' as flow_in_id from dual )tmp;
drop table if exists EdgeDensity_func_test_result;
create table EdgeDensity_func_test_result (   node1 string,   node2 string,   node1_edge_cnt bigint,   no
de2_edge_cnt bigint,   triangle_cnt bigint,   density double );
```

Graph structure shows the graph structure.

Graph structure



Output

```
1,2,4,4,2,0.5
2,3,4,4,3,0.75
2,5,4,7,3,0.75
3,1,4,4,2,0.5
3,4,4,4,2,0.5
4,2,4,4,2,0.5
4,5,4,7,3,0.75
5,1,7,4,3,0.75
5,3,7,4,3,0.75
5,6,7,3,2,0.66667
5,8,7,3,2,0.66667
6,7,3,3,1,0.33333
7,1,3,4,1,0.33333
7,5,3,7,2,0.66667
8,4,3,4,1,0.33333
8,6,3,3,1,0.33333
```

PAI command

```
pai -name EdgeDensity
-project algo_public
-DinputEdgeTableName=EdgeDensity_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=EdgeDensity_func_test_result;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-

Parameter	Description	Required	Default value
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.10.10. Counting triangle

All triangles can be output to an undirected graph G.

Parameter settings

maxEdgeCnt: Optional. If the node degree is larger than the value of this parameter, sampling is required. Default value: 500.

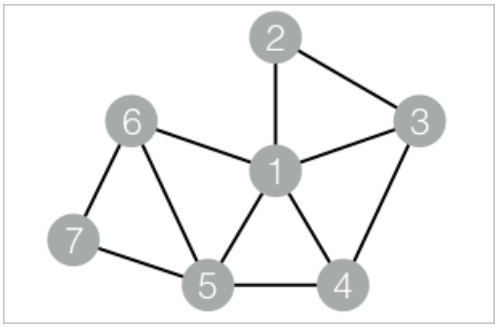
Examples - Testing data

SQL statement to generate data:

```
drop table if exists TriangleCount_func_test_edge;
create table TriangleCount_func_test_edge
as select * from (
select '1' as flow_out_id,
'2' as flow_in_id from dual   union all
select '1' as flow_out_id,
'3' as flow_in_id from dual   union all
select '1' as flow_out_id,
'4' as flow_in_id from dual   union all
select '1' as flow_out_id,
'5' as flow_in_id from dual   union all
select '1' as flow_out_id,
'6' as flow_in_id from dual   union all
select '2' as flow_out_id,
'3' as flow_in_id from dual   union all
select '3' as flow_out_id,
'4' as flow_in_id from dual   union all
select '4' as flow_out_id,
'5' as flow_in_id from dual   union all
select '5' as flow_out_id,
'6' as flow_in_id from dual   union all
select '5' as flow_out_id,
'7' as flow_in_id from dual   union all
select '6' as flow_out_id,
'7' as flow_in_id from dual )tmp;
drop table if exists TriangleCount_func_test_result;
create table TriangleCount_func_test_result (   node1 string,   node2 string,   node3 string );
```

Graph structure shows the graph structure.

Graph structure



Output

```
1,2,3
1,3,4
1,4,5
1,5,6
5,6,7
```

PAI command

```
pai -name TriangleCount
-project algo_public
-DinputEdgeTableName=TriangleCount_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=TriangleCount_func_test_result;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
maxEdgeCnt	If the node degree is larger than the value of this parameter, sampling is required.	No	500
workerNum	The number of workers.	No	-

Parameter	Description	Required	Default value
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.10.11. Tree depth

In a tree network, this component outputs the depth of each node in a tree and the tree ID.

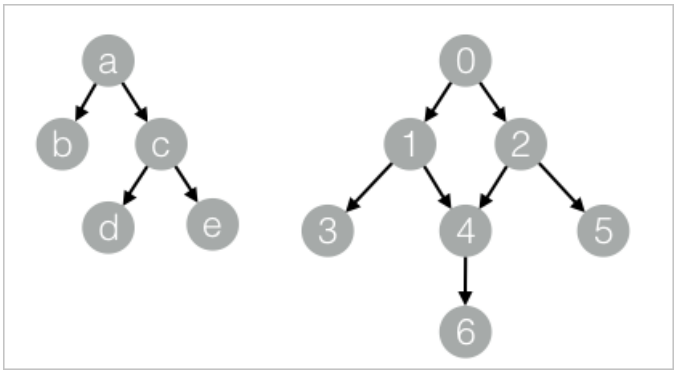
Examples - Testing data

SQL statement to generate data:

```
drop table if exists TreeDepth_func_test_edge;
create table TreeDepth_func_test_edge
as select * from (
select '0' as flow_out_id,
'1' as flow_in_id from dual    union all
select '0' as flow_out_id,
'2' as flow_in_id from dual    union all
select '1' as flow_out_id,
'3' as flow_in_id from dual    union all
select '1' as flow_out_id,
'4' as flow_in_id from dual    union all
select '2' as flow_out_id,
'4' as flow_in_id from dual    union all
select '2' as flow_out_id,
'5' as flow_in_id from dual    union all
select '4' as flow_out_id,
'6' as flow_in_id from dual    union all
select 'a' as flow_out_id,
'b' as flow_in_id from dual    union all
select 'a' as flow_out_id,
'c' as flow_in_id from dual    union all
select 'c' as flow_out_id,
'd' as flow_in_id from dual    union all
select 'c' as flow_out_id,
'e' as flow_in_id from dual )tmp;
drop table if exists TreeDepth_func_test_result;
create table TreeDepth_func_test_result (   node string,   root string,   depth bigint );
```

Graph structure shows the graph structure.

Graph structure



Output

```
0,0,0
1,0,1
2,0,1
3,0,2
4,0,2
5,0,2
6,0,3
a,a,0
b,a,1
c,a,1
d,a,2
e,a,2
```

PAI command

```
pai -name TreeDepth
-project algo_public
-DinputEdgeTableName=TreeDepth_func_test_edge
-DfromVertexCol=flow_out_id
-DtoVertexCol=flow_in_id
-DoutputTableName=TreeDepth_func_test_result;
```

Algorithm parameters

Parameters

Parameter	Description	Required	Default value
inputEdgeTableName	The name of the input edge table.	Yes	-
inputEdgeTablePartitions	The partitions selected from the input edge table.	No	The whole table is selected by default.

Parameter	Description	Required	Default value
fromVertexCol	The start vertex column in the input edge table.	Yes	-
toVertexCol	The end vertex column in the input edge table.	Yes	-
outputTableName	The name of the output table.	Yes	-
outputTablePartitions	The partitions in the output table.	No	-
lifecycle	The lifecycle of the output table.	No	-
workerNum	The number of workers.	No	-
workerMem	The memory size per worker.	No	4096
splitSize	The data split size.	No	64

4.4.11. Tools

4.4.11.1. SQL script

You can use the SQL script editor to write SQL statements.

1. Drag and drop the **SQL Script** component onto the canvas.
2. Connect the input table to the **SQL Script** component, and then click **SQL Script**. The following configuration pane is displayed.
3. Write an SQL script in the text box.
   - An SQL script supports one to four inputs and one output.
   - You can write only one SQL statement.
   - The input data is automatically mapped to tables t1 through t4. You can directly call ${t1}, ${t2}, ${t3}, and ${t4} without specifying the table names.
   - The sample SQL script calculates the number of rows in the input table.

4.4.12. Financials

4.4.12.1. Binning

The Binning component performs data binning based on equal-width or equal-frequency.

PAI command

```
PAI -name binning
    -project algo_public
    -DinputTableName=input
    -DoutputTableName=output
```

Parameters

Parameter	Description	Valid value	Default
inputTableName	Required. The name of the input table.	Table name	N/A
outputTableName	Required. The name of the output table.	Table name	N/A
selectedColNames	Optional. The names of columns selected from the input table for data binning.	Column name	All columns are selected, except for the label column.
labelColumn	Optional. The name of the column that stores the labels.	Column name	No label column is selected.
validTableName	Required when the binning mode (binningMethod) is set to auto. The name of the table used for calibration.	Table name	No table is specified for calibration.
validTablePartitions	Optional. The partitions selected from the calibration table.	Partition name	The whole table is selected.
inputTablePartitions	Optional. The partitions selected from the input table.	Partition name	The whole table is selected.
inputBinTableName	Optional. The name of the input binning table.	Table name	No binning table is specified.
selectedBinColNames	Optional. The names of the columns selected from the binning table.	Column name	No column is selected.

Parameter	Description	Valid value	Default
positiveLabel	Optional. The value used to represent positive samples.	N/A	1
nDivide	Optional. The number of bins.	A positive integer	10
colsNDivide	Optional. The numbers of bins customized for specified columns. Example: col0:3,col2:5. Columns specified in the colsNDivide parameter but not included in the selectedColNames parameter are also processed by the Binning component. For example, if selectedColNames is set to col0,col1 and colsNDivide is set to col0:3,col2:5, data binning is perform based on col0:3,col1:10,col2:5.	N/A	No custom binning rule is specified.
isLeftOpen	Optional. The type of the interval, which can be left-open, right-closed or left-closed, right-open.	true and false	true
stringThreshold	Optional. The discrete value threshold. Values below this threshold are put into other bins.	N/A	No discrete value threshold is set.
colsStringThreshold	Optional. The thresholds for specified columns. Specify the values in the same format of the colsNDivide parameter.	N/A	No threshold is set.

Parameter	Description	Valid value	Default
binningMethod	Optional. The binning mode.	quantile (equal-frequency), bucket (equal-width), and auto (automatic binning). If you select auto, monotonic binning is used based on equal-frequency.	quantile
lifecycle	Optional. The lifecycle of the output table.	A positive integer	No lifecycle is set.
coreNum	Optional. The number of cores.	A positive integer	Automatically calculated.
memSizePerCore	Optional. The memory size per core.	A positive integer	Automatically calculated.

Constraints

Binning constraints must be used together with the Scorecard Training component. You can add constraints to the weight of each dummy variable when the Binning component discretizes features and transforms them into dummy variables. The definitions of the constraints are as follows:

- Ascending order: adds weights to the dummy variables of a feature based on the index values in ascending order. This means that dummy variables with higher index values have larger weights.
- Descending order: adds weights to the dummy variables of a feature based on the index values in descending order. This means that dummy variables with lower index values have lighter weights.
- Same weight: the weights of two dummy variables of a feature must be the same.
- Set weight to 0: sets the weight of a dummy variable to 0.
- Set weight to a specified value: sets the weight of a dummy variable to a floating point value.
- WOE order: adds weights to the dummy variables of a feature based on the WOE values in ascending order. This means that dummy variables with higher WOE values have larger weights.

4.4.12.2. Data conversion

Parameters

Parameter	Description
inputFeatureTableName	Required. The name of the input feature table.
inputBinTableName	Required. The name of the binning result table.

Parameter	Description
inputFeatureTablePartitions	Optional. The partitions selected from the feature table. By default, all partitions are selected.
outputTableName	Required. The name of the output table.
featureColNames	Optional. The names of the features selected from the input feature table. By default, all columns are selected.
metaColNames	Optional. The names of the columns to be reserved in the output table without data conversion. By default, no column is reserved. You can specify columns such as label and sample_id.
transformType	Optional. The type of data conversion. Valid values: normalize (normalization), dummy (discretization), and woe (WOE transformation). The default is dummy.
itemDelimiter	Optional. The delimiter used to separate features. By default, commas (,) are used. Only data discretization supports this parameter.
kvDelimiter	Optional. The delimiter used to separate keys and values. By default, colons (:) are used. Only data discretization supports this parameter.
lifecycle	Optional. The lifecycle of the output table. By default, no lifecycle is set.
coreNum	Optional. The number of cores. By default, the number of cores is automatically calculated.
memSizePerCore	Optional. The memory size per core in megabytes. By default, the memory size per core is automatically calculated.

instance

PAI command

```
PAI -name data_transform
-project algo_public
-DinputFeatureTableName=feature_table
-DinputBinTableName=bin_table
-DoutputTableName=output_table
-DmetaColNames=label
-DfeatureColNames=feaname1,feaname2
```

Normalization

The Normalization component transforms variable values to a scale of 0 to 1. Missing values are imputed with zeros. The algorithm is as follows:

```
if feature_raw_value == null or feature_raw_value == 0 then
   feature_norm_value = 0.0
else
   bin_index = FindBin(bin_table, feature_raw_value)
   bin_width = round(1.0 / bin_count * 1000) / 1000.0
   feature_norm_value = 1.0 - (bin_count - bin_index - 1) * bin_width
```

Output format

Normalization and WOE conversion tables output regular tables.

Dummy variable conversion in discretization outputs a table that contains KV pairs. The output variables are in the format of [${feaname}]\_bin\_${bin_id}. Taking variable sns as an example:

- If the variable sns is put into the second bin, the output variable is `[sns]_bin_2`.
- If the variable sns does not have a value, it is put into an empty bin. The output variable is `[sns]_bin_null`.
- If the variable sns has a value but it cannot be put into any of the predefined bins, it is put into the else bin. The output variable is `[sns]_bin_else`.

4.4.12.3. Scorecard training

The scorecard is a modeling tool widely used for credit score evaluation. It uses binning to discretize variables, and then uses linear models (logistic regression or linear regression) to train a model. The model training process includes feature selection and score transformation. The scorecard also allows you to add constraints to the variables during model training.

> **Note** If you use the scorecard without binning, the entire model training process is equivalent to logistic regression or linear regression.

Feature engineering

The main difference between the scorecard and normal linear models is that the scorecard performs feature engineering before it trains linear models. The Scorecard component provides two methods for feature engineering. You must use binning to discretize the features first no matter which method you choose. One of the methods is to use one hot encoding to encode the variable binning results, and then generate N dummy variables. N represents the number of bins. The other method is to use Weight of Evidence (WOE) transformation. It replaces the original value of a variable with the WOE value of the bin where the variable is placed.

> **Note** You can add constraints for each variable when you transform variables to dummy variables.

Score transformation

In the credit score evaluation scenario, you must use linear transformation to transform the odds of the samples in the prediction results to credit scores as follows:

```
log(odds) = \sum(wx) = a scaled_score + b
```

The parameters in the formula are as follows:

- scaledValue: specifies a base point to be scaled.
- odds: specifies an odd level.
- pdo: specifies the points to double the odds.

For example, if the scaledValue, odds, and pdo parameters are set to 800, 50, and 25, the two vectors that determine the score scale are as follows:

```
log(40) = a * 800 + b
log(80) = a * 825 + b
```

Calculate the values of a and b, and then perform a linear transformation to obtain the scores.

The scaling information is specified in JSON format by using the -Dscale parameter as follows:

```
{"scaledValue":800,"odds":50,"pdo":25}
```

The three parameters must be set at the same time.

Add constraints

You can add constraints for variables during scorecard training. You can set the score of a bin to a fixed value, set a proportion between the scores of two bins, or limit the scores of bins. For example, you can sort the scores of the bins by WOE. The implementation of constraints depends on the underlying optimization algorithms with constraints. You can set a constraint in the Binning component. After you set the parameters, the component generates a constraint in JSON format, and then passes the constraint to the component connected to it. The supported constraints are as follows:

- <: sorts the weights of the variables in ascending order.
- >: sorts the weights of the variables in descending order.
- =: sets the weights of the variables to a fixed value.
- %: sets a proportion between the weights of two variables.
- UP: sets an upper limit for the weights of the variables.
- LO: sets a lower limit for the weights of the variables.

A constraint is stored in a table as a JSON string. The table contains only one row and one column.

```
{
  "name": "feature0",
  "<": [
    [0,1,2,3]
  ],
  ">": [
    [4,5,6]
  ],
  "=": [
    "3:0","4:0.25"
  ],
  "%": [
    ["6:1.0","7:1.0"]
  ]
}
```

Built-in constraints

Every original variable has a default constraint. The average score of the population in a variable must be 0. Based on this constraint, the scaled_weight in the intercept options of the scorecard model equals the average score of the entire population.

Optimization algorithms

In the advanced options of the Scorecard component, you can select optimization algorithms to be used in model training. The supported optimization algorithms are as follows:

- L-BFGS
- Newton's Method
- Barrier Method
- SQP

L-BFGS is a first-order optimization algorithm for processing large amounts of feature data. Newton's Method is a classic tier-2 optimization algorithm. It is fast in regression and accurate. However, it is not suitable for processing large amounts of feature data because it needs to calculate the second-order Hessian Matrix. The two algorithms do not have any constraints. When these algorithms are selected, the system automatically ignores the constraints.

If you do not want the system to ignore the constraints, select Barrier Method or SQP. Barrier Method and SQP are second-order optimization algorithms. When no constraint is set, they are equivalent to Newton's Method. Barrier Method and SQP have minor differences in performance and accuracy. We recommend that you choose SQP. If you are not familiar with optimization algorithms, we recommend that you choose the Auto-selected by default option. The system will automatically select an optimization algorithm based on the amount of the data and the constraints.

Feature selection

The Scorecard component supports stepwise feature selection. Stepwise is a combination of forward selection and backward selection. Each time the system selects a new variable by forward selection and adds it to the model, it must perform a backward selection. The backward selection starts with the variables in the model, and eliminates the ones with significance not meeting the requirements. Stepwise feature selection supports various types of target functions and feature transformation methods. Therefore, stepwise feature selection also supports multiple selection standards. Currently, the following standards are supported:

- Marginal contribution: It can be applied to all target functions and feature engineering methods.
- Score test: It only supports WOE transformation and logistic regression without feature engineering.
- F test: It only supports WOE transformation and linear regression without feature engineering.

Marginal contribution

The marginal contribution is the difference between the target functions of Model A without Variable X and target functions of Model B with Variable X after both models are trained. It is the marginal contribution of Variable X to all the other variables in Model B. In the scenario of transforming variables to dummy variables by feature engineering, the marginal contribution of Variable X is the difference between the target functions of all dummy variables in Model A without Variable X and target functions of all dummy variables in Model B with Variable X. Therefore, using marginal contribution to select features is supported by all feature engineering methods.

Marginal contribution makes feature selection more open-ended. It is not restricted to a certain type of models. Only variables that contribute to the target functions are passed to the model. Marginal contribution has certain disadvantages when compared with statistical significance. Typically, statistical significance chooses 0.05 as its threshold. Marginal contribution does not provide a recommended threshold for beginners. We recommend that you set the threshold to 10E-5.

Score test

Score test is only suitable for feature selection in logistic regression. During a forward selection, a model with only intercept options is trained first. In each subsequent regression, the score chi-squares of the variables that have not been passed to the model are calculated. The variable with the largest score chi-square is passed to the model. The P-value corresponds to the largest score chi-square is also calculated based on chi-square distribution. If the P-value is greater than the given SLENTRY value, feature selection is complete.

After the forward selection is complete, a backward selection is performed for the variable passed to the model. The Wald chi-square of the variable and the corresponding P-value are calculated. If the P-value is greater than the given SLSTAY value, the variable is removed from the model. The system then starts a new regression.

F test

F test is only suitable for feature selection in linear regression. During a forward selection, a variable with only intercept options is trained first. In each subsequent regression, the F-values of the variables that have not been passed to the model are calculated. F-value calculation is similar to marginal contribution calculation. Both of them need to train two models to calculate the F-value of a variable. The F-value fits the F distribution. The corresponding P-value can be calculated based on the probability density function of the F distribution. If the P-value is greater than the given SLENTRY value, the variable is not passed to the model, and the forward selection is complete.

The backward selection process uses the F-value to calculate the significance of the variable in a way similar to a score test.

Forcibly selected variables

Before you perform feature selection, you can specify variables to be forcibly passed to the model. The specified variables are passed to the model regardless of their significance. No forward selection or backward selection is performed for these variables.

The number of regressions and significance levels (SLENTRY and SLSTAY) are defined by using a JSON string in the -Dselected parameter as follows:

```
{"max_step":2, "slentry": 0.0001, "slstay": 0.0001}
```

If the -Dselected parameter is left empty or max_step is set to 0, no feature selection is performed.

Model report

The Scorecard component outputs data to a model report. The model report contains basic model evaluation statistics, such as the binning information, binning constraints, WOE values, and marginal contribution information. The following table lists the fields contained in the model report:

Field	Type	Description
feaname	string	The name of the feature.
binid	bigint	The ID of a bin.
bin	string	The description of the bin, which indicates the interval of the bin.
constraint	string	The constraints of the bin specified for model training.
weight	double	The weight of a binning variable. For a non-scorecard model without binning, this field indicates the weight of a model variable.

Field	Type	Description
scaled_weight	double	For score transformation in scorecard training, this field indicates the score linearly transformed from the weight of a binning variable.
woe	double	A statistical indicator. It indicates the WOE value of a bin in the training set.
contribution	double	A statistical indicator. It indicates the marginal contribution value of a bin in the training set.
total	bigint	A statistical indicator. It indicates the total number of samples in a bin in the training set.
positive	bigint	A statistical indicator. It indicates the number of positive samples in a bin in the training set.
negative	bigint	A statistical indicator. It indicates the number of negative samples in a bin in the training set.
percentage_pos	double	A statistical indicator. It indicates the proportion between positive samples in a bin and total positive samples in the training set.
percentage_neg	double	A statistical indicator. It indicates the proportion between negative samples in a bin and total negative samples in the training set.
test_woe	double	A statistical indicator. It indicates the WOE value of a bin in the testing set.
test_contribution	double	A statistical indicator. It indicates the marginal contribution value of a bin in the testing set.

Field	Type	Description
test_total	bigint	A statistical indicator. It indicates the total number of samples in a bin in the testing set.
test_positive	bigint	A statistical indicator. It indicates the number of positive samples in a bin in the testing set.
test_negative	bigint	A statistical indicator. It indicates the number of negative samples in a bin in the testing set.
test_percentage_pos	double	A statistical indicator. It indicates the proportion between positive samples in a bin and total positive samples in the testing set.
test_percentage_neg	double	A statistical indicator. It indicates the proportion between negative samples in a bin and total negative samples in the testing set.

Algorithm parameters

Parameter	Description	Valid value	Default
inputTableName	Required. The input table that contains features.	N/A	N/A
inputTablePartitions	Optional. The partitions selected from the input table.	N/A	The whole table is selected.
inputBinTableName	Optional. The binning result table. If this parameter is set, the system automatically discretizes the original features based on the binning rules in the binning result table.	N/A	N/A

Parameter	Description	Valid value	Default
featureColNames	Optional. The names of feature columns selected from the input table.	N/A	All columns in the input table are selected by default, except for the label column.
labelColName	Required. The names of the target columns.	N/A	N/A
outputTableName	Required. The name of the output table.	N/A	N/A
inputConstraintTableName	Optional. A constraint. The constraint is a JSON string stored in a cell of a table.	N/A	
optimization	Optional. The optimization algorithm.	lbfgs, newton, barrier_method, sqp, and auto. Currently, only the sqp and barrier_method algorithms support constraints. If you set the value to auto, the system automatically selects an optimization algorithm based on the input data and corresponding parameters. We recommend that you set the value to auto if you are unfamiliar with the listed optimization algorithms.	auto
loss	Optional. The type of the loss function.	logistic_regression and least_square.	logistic_regression
iterations	Optional. The maximum number of regressions.	N/A	100
l1Weight	Optional. The weight of the L1 regularization parameter. Currently, only lbfgs supports l1weight.	N/A	0

Parameter	Description	Valid value	Default
l2Weight	Optional. The weight of the L2 regularization parameter.	N/A	0
m	Optional. The number of regressions performed by L-BFGS. Only L-BFGS supports this parameter.	N/A	10
scale	Optional. The weight scaling information of the scorecard.	N/A	Null
selected	Optional. Feature selection in scorecard training.	N/A	Null
convergenceTolerance	Optional. The convergence tolerance.	N/A	1e-6
positiveLabel	Optional. The category of positive samples.	N/A	1
lifecycle	Optional. The lifecycle of the output table.	N/A	No lifecycle is set.
coreNum	Optional. The number of vCores.	N/A	Automatically calculated.
memSizePerCore	Optional. The memory size per core.	N/A	Automatically calculated.

4.4.12.4. Scorecard prediction

The Scorecard Prediction component predicts credit scores based on the input data. It uses a model generated by a model training component. Supported model training components include the Scorecard Training, Logistic Regression for Binary Classification (in the Financials folder), and Linear Regression (in the Financials folder) components.

Input parameters

The Scorecard Prediction component has the following parameters:

- **Feature Column**: specifies the feature columns to be used for predicting credit scores. By default, all columns are selected.
- **Columns Reserved in Result Table**: specifies the columns to be appended to the prediction result table without any changes, such as the ID column and target column.

- **Output Variable Score**: specifies whether to output the score of each variable. The final score equals the score in the intercept option plus the scores of all variables.

Score table

The following is an example of the score table output by the component.

The first column churn is the column appended to the result table from the input table. The data in this column does not affect the prediction results. The remaining three columns display the prediction results. The definitions of these columns are as follows:

Column name	Type	Description
prediction_score	Double	The predicted scores column. In a linear model, the feature values and model weight values are summed up or multiplied to obtain the predicted scores. In a scorecard model, if score transformation is performed, the transformed scores are input into this column.
prediction_prob	Double	The probability values of positive samples in binary classification. The probability values are transformed from the original scores (before score transformation) by using the sigmoid function.
prediction_detail	String	The probability values of positive and negative samples described in JSON strings. Value 0 represents negative and value 1 represents positive. Example: { “0” :0.1813110520,” 1” :0.8186889480}.

PAI command

```
pai -name=lm_predict
    -project=algo_public
    -DinputFeatureTableName=input_data_table
    -DinputModelTableName=input_model_table
    -DmetaColNames=sample_key,label
    -DfeatureColNames=fea1,fea2
    -DoutputTableName=output_score_table
```

Algorithm parameters

Parameter	Description	Valid value	Default
inputFeatureTableName	Required. The name of the input table that stores feature data.	N/A	N/A
inputFeatureTablePartitions	Optional. The partitions selected from the input table.	N/A	The whole table is selected.
inputModelTableName	Required. The name of the model table.	N/A	N/A
featureColNames	Optional. The names of the feature columns selected from the input table.	N/A	All columns are selected.
metaColNames	Optional. The names of the columns to be reserved in the result table.	N/A	The meta column is excluded. You can specify columns such as label and sample_id.
outputFeatureScore	Optional. It specifies whether to output variable scores to the result table.	true and false	false
outputTableName	Required. The name of the result table.	N/A	N/A
lifecycle	Optional. The lifecycle of the result table.	N/A	No lifecycle is set.
coreNum	Optional. The number of cores.	N/A	Automatically calculated.
memSizePerCore	Optional. The memory size per core.	N/A	Automatically calculated.

4.4.12.5. PSI

Population stability index (PSI) is an important metric to identify a shift in two samples of a population. For example, you can use it to measure whether the changes in the population within two months are stable. A PSI value smaller than 0.1 indicates insignificant changes. A PSI value between 0.1 and 0.25 indicates minor changes. A PSI value greater than 0.25 indicates major changes in the population.

When the changes in a population over time are unstable, you can use charts to identify the changes. You can use binning to discretize variables into multiple bins, calculate the number and proportion of the samples in each bin, and then display the statistics in a chart, as shown in the following figure.

This method can directly show whether a variable in two samples changes significantly. However, the shift in these changes cannot be measured by using this method. This means that the population stability cannot be automatically monitored. To resolve this issue, you can use the PSI component. Before you use the PSI component, you must use the Binning component to discretize the data. The formula for calculating PSI values is as follows:

Examples

The following figure shows a use case of the PSI component. The PSI component is connected to the Binning component and two sample datasets. You only need to specify the columns for PSI calculation in the PSI component.

Result

PAI command

```
PAI -name psi
-project algo_public
-DinputBaseTableName=psi_base_table
-DinputTestTableName=psi_test_table
-DoutputTableName=psi_bin_table
-DinputBinTableName=pai_index_table
-DfeatureColNames=fea1,fea2,fea3
-Dlifecycle=7
```

Algorithm parameters

Parameter	Description	Valid value	Default
inputBaseTableName	Required. The name of the base table. The shift of the population is calculated based on the samples in the base and test tables.	N/A	N/A
inputBaseTablePartitions	Optional. The partitions in the base table.	N/A	The whole table is selected.
inputTestTableName	Required. The name of the test table. The shift of the population is calculated based on the samples in the base and test tables.	N/A	N/A
inputTestTablePartitions	Optional. The partitions in the test table.	N/A	The whole table is selected.

Parameter	Description	Valid value	Default
inputBinTableName	Required. The name of the binning result table.	N/A	N/A
featureColNames	Optional. The features specified for PSI calculation.	N/A	All features are selected.
outputTableName	Required. The name of the output table (PSI statistics).	N/A	N/A
lifecycle	Optional. The lifecycle of the output table.	N/A	No lifecycle is set.
coreNum	Optional. The number of cores.	N/A	Automatically calculated.
memSizePerCore	Optional. The memory size per core.	N/A	Automatically calculated.

4.5. OpenAPI

4.5.1. Query PMML models

Operation name

ListPMMLModels

Description

You can call this operation to query PMML models by project ID, experiment ID, or owner.

Request parameters

Parameter	Type	Required	Example	Description
OnwerId	String	Yes	11769368777159105	The UID of a model owner.
ProjectId	Long	Yes	10009	The ID of a project.
ExperimentId	Long	No	4294	The ID of an experiment.

Parameter	Type	Required	Example	Description
Action	String	Yes	ListPMMLModels	The operation that you want to perform. Set the value to ListPMMLModels.

Response parameters

Parameter	Type	Description
Experiments	List<ExperimentModelInfo>	An array of experiment properties returned.
Experiment	ExperimentModelInfo	The returned experiment information.
ExpId	Long	The ID of the experiment to which the models belong.
Models	List<ModelInfo >	An array of model properties returned.
Model	ModelInfo	The returned model information.
Name	String	The display name of the model.
Owner	String	The ID of the model owner.
Description	String	The description of the model.
ModelId	String	The ID of the model.
ModelName	String	The name of the underlying model generated by the corresponding algorithm.
CreateTime	Date	The time when the model was generated.
ExperimentId	String	The ID of the experiment from which the model is generated.
UpdateTime	Date	The time when the model was updated.
Project	String	The name of the project to which the model belongs.
ProjectId	String	The ID of the project to which the model belongs.

Action	String	The name of the API operation.
AccessKeyId	String	The AccessKey ID provided to you by Alibaba Cloud.
Signature	String	The signature string.
SignatureMethod	String	The signing method.
SignatureVersion	String	The version of the signature encryption algorithm.
SignatureNonce	String	A unique, random number used to prevent replay attacks.
Timestamp	String	The timestamp of the request.
Version	String	The version number of the API. The value must be in the YYYY-MM-DD format.
Format	String	The language of the response.

Sample requests

```
http://pop.pai.idst.inter.env8d.com/?Action=ListPMMLModels&ProjectId=10009&Version=2019-09-25&ExperimentId=4294&OnwerId=11769368777159105&<Common request parameters>
```

Sample responses

```
<ListPMMLModelsResponse>
<Experiments>
<Experiment>
<ExpId>4294</ExpId>
<Models>
<Model>
<Name>Logistic regression for binary classification-1-Model</Name>
<Owner>11769368777159105</Owner>
<Description>Logistic regression for binary classification-1-Model</Description>
<ModelId>6119</ModelId>
<ModelName>xlab_m_logisticregressi_51466_v0</ModelName>
<CreateTime>2019-09-23 17:06:42</CreateTime>
<ExperimentId>4294</ExperimentId>
<UpdateTime>2019-09-26 19:33:03</UpdateTime>
<Project>pai_emr</Project>
<ProjectId>10009</ProjectId>
</Model>
```

```
<Model>
<Name>Random forest1-AUC-0</Name>
<Owner>11769368777159105</Owner>
<Description>Random forest1-AUC-0</Description>
<ModelId>6200</ModelId>
<ModelName>xlab_m_random_forests_1_51463_v0_m_0</ModelName>
<CreateTime>2019-09-26 12:38:12</CreateTime>
<ExperimentId>4294</ExperimentId>
<UpdateTime>2019-09-26 12:38:12</UpdateTime>
<Project>pai_emr</Project>
<ProjectId>10009</ProjectId>
</Model>
</Models>
</Experiment>
</Experiments>
<RequestId>0a948186156950036564347 43d0059</RequestId>
<ErrMsg>Successful</ErrMsg>
<ErrCode>success</ErrCode>
</ListPMMLModelsResponse>{
 "ListPMMLModelsResponse": {
  "Experiments": {
   "Experiment": {
    "ExpId": "4294",
    "Models": {
     "Model": [
      {
       "Name": "Logistic regression for binary classification-1-Model",
       "Owner": "11769368777159105",
       "Description": "Logistic regression for binary classification-1-Model",
       "ModelId": "6119",
       "ModelName": "xlab_m_logisticregressi_51466_v0",
       "CreateTime": "2019-09-23 17:06:42",
       "ExperimentId": "4294",
       "UpdateTime": "2019-09-26 19:33:03",
       "Project": "pai_emr",
       "ProjectId": "10009"
      },
      {
       "Name": "Random forest1-AUC-0",
       "Owner": "11769368777159105",
```

```
            "Description": "RandomForest1-AUC-0",
            "ModelId": "6200",
            "ModelName": "xlab_m_random_forests_1_51463_v0_m_0",
            "CreateTime": "2019-09-26 12:38:12",
            "ExperimentId": "4294",
            "UpdateTime": "2019-09-26 12:38:12",
            "Project": "pai_emr",
            "ProjectId": "10009"
          }
        ]
      }
    }
  },
  "RequestId": "0a948186156950036564347 43d0059",
  "ErrMsg": "Successful",
  "ErrCode": "success"
 }
}
```


4.5.2. Query detailed information about a PMML model

Operation name

DescribePMMLMode

Description

You can call this operation to query detailed information about a model.

Request parameters

Parameter	Type	Required	Example	Description
ModelId	Integer	Yes	6200	The ID of the model.
Action	String	Yes	DescribePMMLMode	The operation that you want to perform. Set the value to DescribePMMLMode.

Response parameters

Parameter	Type	Description
Models	List<ModelInfo >	An array of model properties returned.
Model	ModelInfo	Detailed information about the model.
Name	String	The display name of the model.
Owner	String	The ID of the model owner.
Description	String	The description of the model.
ModelId	String	The ID of the model.
ModelName	String	The name of the underlying model generated by the algorithm.
CreateTime	Date	The time when the model was generated.
ExperimentId	String	The ID of the experiment from which the model is generated.
UpdateTime	Date	The time when the model was updated.
Project	String	The name of the project to which the model belongs.
ProjectId	String	The ID of the project to which the model belongs.

Sample requests

```
http://pop.pai.idst.inter.env8d.com/?Action=DescribePMMLMode&ModelId=6200&<Common request parameters>
```

Sample responses

<DescribePMMLModeResponse>
<Data>
<ModelInfo>
<Name>Logistic regression for binary classification-1-Model</Name>
<Owner>11769368777159105</Owner>
<ModelId>6119</ModelId>
<Description>Logistic regression for binary classification-1-Model</Description>
<ModelName>xlab_m_logisticregressi_51466_v0</ModelName>
<CreateTime>2019-09-23 17:06:42</CreateTime>
<UpdateTime>2019-09-26 19:33:03</UpdateTime>
<ExperimentId>4294</ExperimentId>
<Project>pai_emr</Project>
<ProjectId>10009</ProjectId>
</ModelInfo>
</Data>
<RequestId>0a9441531569499268686 8173d0065</RequestId>
<ErrMsg>Successful</ErrMsg>
<ErrCode>success</ErrCode>
</DescribePMMLModeResponse>{
"DescribePMMLModeResponse": {
"Data": {
"ModelInfo": {
"Name": "Logistic regression for binary classification-1-Model",
"Owner": "11769368777159105",
"ModelId": "6119",
"Description": "Logistic regression for binary classification-1-Model",
"ModelName": "xlab_m_logisticregressi_51466_v0",
"CreateTime": "2019-09-23 17:06:42",
"UpdateTime": "2019-09-26 19:33:03",
"ExperimentId": "4294",
"Project": "pai_emr",
"ProjectId": "10009"
}
},
"RequestId": "0a9441531569499268686 8173d0065",
"ErrMsg": "Successful",
"ErrCode": "success"
}

4.5.3. Download PMML models

4.5.3.1. Generate a download URL for a model

Operation name

GeneratePMMLModelUrl

Description

You can call this operation to generate a download URL for a PMML model. The API call is executed asynchronously. After you call this operation, the system creates a download URL generation task in the background, and returns the task ID to you. You can then use the task ID to query the status of the task. If the task is successfully executed, the URL (OSS endpoint) of the model is returned.

Request parameters

Parameter	Type	Required	Example	Description
ModelId	Integer	Yes	6200	The ID of the model.
Action	String	Yes	GeneratePMMLModelUrl	The operation that you want to perform. Set the value to GeneratePMMLModelUrl.

Response parameters

Parameter	Type	Description
Data	Data	The returned data.
JobId	String	The ID of the download URL generation task.

Sample requests

```
http://pop.pai.idst.inter.env8d.com/?Action=GeneratePMMLModelUrl&ModelId=6200&<Common request parameters>
```

Sample responses

```
<GeneratePMMLModelUrlResponse>
<Data>
<JobId>
asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c
</JobId>
</Data>
<RequestId>0a9481861569501305435669 7d0059</RequestId>
<ErrMsg>Successful</ErrMsg>
<ErrCode>success</ErrCode>
</GeneratePMMLModelUrlResponse>{
 "GeneratePMMLModelUrlResponse": {
   "Data": {
    "JobId": "
asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c
"
   },
   "RequestId": "0a9481861569501305435669 7d0059",
   "ErrMsg": "Successful",
   "ErrCode": "success"
 }
}
```


4.5.3.2. Query a model URL generation task

Operation name

QueryAsynJobStatus

Description

You can call this operation to query the status of a URL generation task. If the task is successfully executed, the URL (OSS endpoint) of the model is returned.

Request parameters

Parameter	Type	Required	Example	Description

Parameter	Type	Required	Example	Description
JobId	String	Yes	asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c	The ID of the task to be queried. The task ID is returned after you call the GeneratePMMLModelUrl operation.
Action	String	Yes	QueryAsynJobStatus	The operation that you want to perform. Set the value to QueryAsynJobStatus.

Response parameters

Parameter	Type	Description
Data	Data	The returned data.
Status	String	The status of the task. Valid values: done, failed, and running.
Info	String	The URL of the model is returned if the task is successfully executed. An error message is returned if the system fails to execute the task.

Sample requests

```
http://pop.pai.idst.inter.env8d.com/?Action=QueryAsynJobStatus&JobId=asynUploadModel2Oss_395fd769-1568-4015-8aa8-a56b2e0da11c&<Common request parameters>
```

Sample responses

```
<QueryAsynJobStatusResponse>
<Data>
<Status>done</Status>
<Info>
<![ CDATA[
http://pai-global-oss.oss-cn-qingdao-env8d-d01-a.intra.env8d.com/xlab_m_random_forests_1_51463_v
0_m_0-%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%971-AUC-0.pmml?Expires=1569501906&OSSAccess
KeyId=yMPdrqaNstzXKsNY&Signature=uRM1OFDZeDNzZo%2BjG87s09uUd6****
]]>
</Info>
</Data>
<RequestId>0a9481861569501342195757 9d0059</RequestId>
<ErrMsg>Successful</ErrMsg>
<ErrCode>success</ErrCode>
</QueryAsynJobStatusResponse>{
 "QueryAsynJobStatusResponse": {
   "Data": {
    "Status": "done",
    "Info": "

http://pai-global-oss.oss-cn-qingdao-env8d-d01-a.intra.env8d.com/xlab_m_random_forests_1_51463_v
0_m_0-%E9%9A%8F%E6%9C%BA%E6%A3%AE%E6%9E%971-AUC-0.pmml?Expires=1569501906&OSSAccess
KeyId=yMPdrqaNstzXKsNY&Signature=uRM1OFDZeDNzZo%2BjG87s09uUd6****

"
   },
   "RequestId": "0a9481861569501342195757 9d0059",
   "ErrMsg": "Successful",
   "ErrCode": "success"
 }
}
```

4.5.4. SDKs

Alibaba Cloud SDKs

```
https://developer.aliyun.com/sdk?spm=5176.10695662.1kquk9v2l.2.1e734735x9Gptc&aly_as=m3lFQpXP<
dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>{$version}</version></dependency>
```

SDK use case

Example: call an RPC API operation.

```
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
public class Sample {
    public static void main(String[] args) {
        // Create a default ACS client and initialize it.
        DefaultProfile profile = DefaultProfile.getProfile(
            "<your-region-id>",         // The region ID.
            "<your-access-key-id>",     // The AccessKey ID.
            "<your-access-key-secret>"); // The AccessKey secret.
        IAcsClient client = new DefaultAcsClient(profile);
        // Create an API request and configure parameters.
        CommonRequest request = new CommonRequest();
        request.setDomain("ecs.aliyuncs.com");
        request.setVersion("2014-05-26");
        request.setAction("DescribeInstanceStatus");
        request.putQueryParameter("PageNumber", "1");
        request.putQueryParameter("PageSize", "30");
        try {
            CommonResponse response = client.getCommonResponse(request);
            System.out.println(response.getData());
        } catch (ServerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClientException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Example: call an RESTful API operation.

```
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.profile.DefaultProfile;
public class Sample {
    public static void main(String[] args) {
        // Create a default ACS client and initialize it.
        DefaultProfile profile = DefaultProfile.getProfile(
            "<your-region-id>",         // The region ID.
            "<your-access-key-id>",     // The AccessKey ID.
            "<your-access-key-secret>"); // The AccessKey secret.
        IAcsClient client = new DefaultAcsClient(profile);
        // Create an API request and configure parameters.
        CommonRequest request = new CommonRequest();
        request.setDomain("cs.aliyuncs.com");
        request.setVersion("2015-12-15");
        request.setUriPattern("/clusters");
        try {
            CommonResponse response = client.getCommonResponse(request);
            System.out.println(response.getData());
        } catch (ServerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ClientException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

SDK parameters

The domain must be in the pop.${pai_domain} format. Set the pai_domain parameter to the domain shown in the Machine Learning Platform for AI.

The version must be 2019-09-25.

The Action string specifies an API operation.

The QueryParameter string specifies API request parameters, excluding the action parameter.

Example:

```
CommonRequest request = new CommonRequest();
request.setDomain("pop.${pai_domain}");
request.setVersion("2019-09-25");
request.setAction("DescribePMMLMode");
request.putQueryParameter("ModelId", "***");
```

4.6. EAS user guide

4.6.1. EAS overview

Elastic Algorithm Service (EAS) allows you to deploy machine learning models, such as PMML, PAI-OfflineModel, TensorFlow, and Caffe models, as services. You can also develop custom online services based on the API standards defined by EAS. You can simply use a JSON file to describe the service that you want to deploy, such as the path of the model, the region where the service is deployed, and the resources used by the service. After the JSON file is prepared, you can use the eascmd client or Alibaba Cloud SDK for Java to deploy it as an online service. The service can be accessed from the production environment and Internet. EAS supports various regions and different types of hardware resources (CPUs and GPUs).

4.6.2. Client

You can use the eascmd client to create, delete, view, and modify services. The eascmd client is stored in the Object Storage Service (OSS) bucket eas-utils. You can use wget or curl to download eascmd.

4.6.3. User authentication

Deployed model services use JSON Web Token (JWT) for user authentication. You can run the following command in eascmd to configure the AccessKey information.

```
./eascmd64 config -i <AccessKeyId> -k <AccessKeySecret> -e {{Domain name of EAS}}
//New users can use the admin account to pass authentication.
./eascmd64 config -i admin -k admin -e {{Domain name of EAS}}
```

4.6.4. Upload files

When you create a service, you must specify the HTTP addresses of the model and processor files. You can store the model and processor files on any HTTP servers.

4.6.5. Create a service

You can run the `create` command to create a service. When you create a service, you must specify the HTTP URLs of the model and processor files to be used by the service. You can upload the files to Object Storage Service (OSS) and specify their OSS paths.

```
eascmd create [service_desc_json]
```

The service_desc_json file contains service information and metadata configurations. The metadata configurations only contain cluster deployment information, such as the region where the cluster is deployed and the required resources. Example:

> **Note** When you run the test command to debug the service, the metadata configurations are ignored.

```
{
 "name": "mnist_saved_model_example",
 "generate_token": "true",
 "model_path": "http://eas-data.oss-cn-shanghai.aliyuncs.com/models%2Fmnist_saved_model.tar.gz",
 "processor": "tensorflow_cpu",
 "metadata": {
  "instance": 1,
  "cpu": 1,
 }
}
```

The parameters in the JSON file are described as follows:

Key	Value
name	The name of the service. Service names are unique inside each region. You can specify the region in the metadata configurations.
generate_token	Specifies whether to generate a token. If the value is set to true, you must add the token to the HTTP header of the request that you send to the service. If the value is set to false, the service is accessible to the public. No authentication is required.
token	Optional. The token used for authenticating requests sent to the service. If you do not specify a token, the generate_token parameter automatically generates a token.
model_path	The path of the model file. For more information, see the note under this table.

Key	Value
model_entry	Optional. The input file of the model. If no input file is specified, the file specified by the model_path parameter is used. You can specify arbitrary files in this parameter. The path of the primary file will be passed to the Load() function in the processor.
model_config	Optional. The configuration of the model. The configuration can be in any text format. The configuration is passed to the second parameter of the LoadWithConfig() function in the processor.
processor	Optional. The built-in processor used to make predictions. When this parameter is specified, the processor_path, processor_entry, processor_mainclass, and processor_type parameters are ignored.
processor_path	Optional. The path of the processor. Only custom processors support this parameter. For more information about processor packages, see the note under this table.
processor_entry	The primary file of the processor. It contains the implementations of the Load() and Process() functions. This parameter is required when you develop a custom processor using the C or C++ language.
processor_mainclass	The primary file of the processor. It defines the main class in Java. This parameter is required when you develop a custom processor using Java.
processor_type	Optional. The language used to develop the processor. Only custom processors support this parameter. Currently, only C++, Java, and Python are supported.
metadata	The metadata of the service.

Note model_path and processor_path specify the inputs of the model and processor in the format of HTTP URLs or OSS paths. When you use the test command to debug the service on your local machine, local paths are supported.

When HTTP URLs are used, the required files must be packaged in the tar.gz, tar.bz2, or ZIP format.

Metadata descriptions

Category	Parameter	Description
Common options	workers	Optional. The number of threads used by each instance to process requests in parallel. The default is 5.
	instance	The number of instances launched by the service.
	CPU	The number of CPUs required by each instance.
	gpu	The number of GPUs required by each instance.
	resource	The name of the resource group. If your service uses CPU resources, ignore this parameter. Supported GPU resource groups are P4_4CORE and P4_8CORE.
Advanced parameter (use with caution)	rpc.batching	Optional. It specifies whether to enable batching at the server end for GPU optimization. The default is false.
	rpc.keepalive	Optional. The maximum amount of time that it takes to process a single request. When the time expires, the server end returns a 408 error and disconnects from the client. The default is 5000 milliseconds.
	rpc.io_threads	Optional. The number of threads used by each instance to handle network inputs and outputs. The default is 4.
	rpc.max_batch_size	Optional. The maximum size of each batch when batching is enabled. The default is 16.
	rpc.max_batch_timeout	Optional. The maximum timeout of each batch when batching is enabled. The default is 50 milliseconds.

Category	Parameter	Description
	rpc.max_queue_size	Optional. The maximum size of the queue. The default is 64. When the queue is full, the server end returns a 450 error and closes the connection. The queue can prevent the server from being overloaded. It also can notify the client to send the request to other instances when the current instance is busy. For a queue that produces a long response time, you can set the size of the queue to a smaller value to prevent large amounts of pending requests from timing out.
	rpc.worker_threads	Optional. The number of threads used by each instance to process requests in parallel. This parameter is the same as the workers parameter. The default is 5.

Example:

```
{
 ...
"metadata": {
  "cpu": 4,
"rpc.max_queue_size": 32,
...
 }
```

```
$ eascmd create pmml.json
[RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DDEB2D
+------------------+-------------------------------------------------------------------------------+
| Intranet Endpoint | http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/savedmodel_exanple |
|          Token | YjQxZDYzZTBiZTZjMzQ5ZmE****                |
+------------------+-------------------------------------------------------------------------------+
[OK] Creating api gateway
[OK] Building image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_cn-shanghai:v0.
0.1-20190224001315]
[OK] Pushing image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_cn-shanghai:v0.
0.1-20190224001315]
[OK] Waiting [Total: 1, Pending: 1, Running: 0]
[OK] Waiting [Total: 1, Pending: 1, Running: 0]
[OK] Service is running
```

4.6.6. Local debugging

Before you deploy the service to the cluster, you can use the local debugging feature to start a local service for debugging. This feature requires a Docker container and Internet access. You must install Docker on the machine where eascmd runs, and then run the following command to debug the service.

```
sudo eascmd test [service_desc_json]
```

Specify the JSON file used to deploy the service in this command. The following example shows how to run this command:

```
[xingke.***@mac-3:~/code/go/src/easgo]$ bin/eascmd test tf.json
[OK] Pulling image: registry.cn-shanghai.aliyuncs.com/eas/eas-worker-amd64:0.1.4
[OK] Pull image done
[OK] Creating container from: registry.cn-shanghai.aliyuncs.com/eas/eas-worker-amd64:0.1.4
[OK] Created container: e39176f85cf41a161bbb2896f76f1c0db0ad4f50e1d78a********
[OK] Serving At: [http://localhost:6942/api/predict/savedmodel_exanple]
[2019-02-24 00:16:27] [172.17.0.2] Fetching model from [http://eas-data.oss-cn-shanghai.aliyuncs.com/models%2Fflypig_scorecard.pmml]
[2019-02-24 00:16:27] [172.17.0.2] Fetching processor from [http://eas-data.oss-cn-shanghai.aliyuncs.com/eas-pmml-processor-0.1-jar-with-dependencies.jar]
[2019-02-24 00:16:28] [172.17.0.2] ----------------------------SERVICE LOG-------------------------------
...
[2019-02-24 00:16:30] [172.17.0.2] [INFO] Token: [WIzMFW5Jb8kckYj****]
...
[2019-02-24 00:16:30] [172.17.0.2] [INFO] Service start successfully
```

In this example, the command starts a local TensorFlow model service. The endpoint of the service is http://localhost:6942/api/predict/savedmodel_exanple.

4.6.7. Modify configurations

You can specify the -D parameter in the modify command to modify the metadata configurations, such as the instance, CPU, and memory configurations.

```
eascmd modify [service_name] -Dmetadata.[attr_name]=[attr_value]
```

For example, you can run the following command to set the number of instances to 10.

```
eascmd modify service_test -Dmetadata.instance=10
```

You can set multiple properties at a time. For example, you can set the number of instances to 10 and the memory size to 2,000 MB.

```
eascmd modify service_test -Dmetadata.instance=10 -Dmetadata.memory=2000
```

4.6.8. Modify a service

You can run the modify command to modify a deployed service.

```
eascmd modify [service_name] -s [service_desc_json]
```

> **Note** You cannot use this command to modify the region where the service is deployed. Note: If you only want to modify the resources used by the service, specify the metadata configurations in the service description file.

4.6.9. Delete a service

You can run the delete command to delete a service.

```
eascmd delete [service_name]
```

When you delete a service, you must specify the service name and the region where the service is deployed.

Examples:

```
$ eascmd delete savedmodel_exanple
Are you sure to delete the service [savedmodel_exanple] in [cn-shanghai]? [Y/n]
[RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DDEB2D
[OK] Service [savedmodel_exanple] in region [cn-shanghai] is terminating
[OK] Service is terminating
[OK] Service is terminating
[OK] Service was deleted successfully
```

4.6.10. Switch service version

You can run the desc command to view the version of the current service and the latest version, and run the version command to switch the service to a version earlier than the latest version.

```
eascmd version [service_name] [version_id]
```

4.6.11. View service list

You can run the list(ls) command to view all services deployed by the current user.

```
$ eascmd ls
[RequestId]: 83945D4EED3E-4D35-A989-831E6BBA39F
+--------------------------+-------------+----------+--------------------+--------------------+---------+--------+-
---------------------------------------+
|     SERVICENAME      |  REGION   | INSTANCE |    CREATETIME     |    UPDATETIME     | STATUS  | WEIGH
T |          SERVICEPATH          |
+--------------------------+-------------+----------+--------------------+--------------------+---------+--------+-
---------------------------------------+
| mnist_saved_model_example | cn-shanghai |       1 | 2019-02-21 16:35:41 | 2019-02-21 16:35:41 | Runnin
g |     0 | /api/predict/mnist_saved_model_example |
```

4.6.12. View service information

You can run the desc command to view detailed information about a deployed service.

```
eascmd desc [service_name]
```

Examples:

```
$ eascmd desc mnist_saved_model_example
+--------------------+-----------------------------------------------------------------------------------------------------
------------+
|            Status | Running                                                                                          |
|       ServiceName | mnist_saved_model_example                                                                                  |
|            Region | cn-shanghai                                                                                      |
|        CreateTime | 2019-02-21 16:35:41                                                                                |
|        UpdateTime | 2019-02-21 16:35:41                                                                                |
|       AccessToken |                                                                                                  |
|      PrivateToken | ZWNjMTNkNDExMmExNjZkYTM4YWQ5YTY****                                                                      |
|     TotalInstance | 1                                                                                                |
|    RunningInstance | 1                                                                                               |
|    PendingInstance | 0                                                                                               |
|             CPU | 1                                                                                                  |
|             GPU | 0                                                                                                  |
|           Memory | 1000M                                                                                             |
|            Image | registry-vpc.cn-shanghai.aliyuncs.com/eas/mnist_saved_model_example_cn-shanghai
:v0.0.1-20190221163541       |
|           Weight | 0                                                                                                |
|      LatestVersion | 1                                                                                              |
|     CurrentVersion | 1                                                                                              |
|           Message | Service start successfully                                                                          |
|      APIGatewayUrl | 1c3b37ea83c047efa0dc6df0cacb70d3-cn-shanghai.alicloudapi.com/EAPI_182848887
9222746_mnist_saved_model_example |
|    APIGatewayAppKey | 25641710                                                                                        |
| APIGatewayAppSecret | 12562a7b8858bbba****                                                                            |
|    IntranetEndpoint | http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/mnist_saved_model_ex
ample                      |
|      ServiceConfig | {                                                                                               |
|                    |   "generate_token": "false",                                                                       |
|                    |   "metadata": {                                                                                 |
|                    |     "cpu": 1,                                                                                   |
|                    |     "instance": 1,                                                                              |
|                    |     "memory": 1000,                                                                             |
```

```
|             |    "region": "cn-shanghai"                                                    |
|             |  },                                                                           |
|             |  "model_path":                                                                |
|             | "http://eas-data.oss-cn-shanghai.aliyuncs.com/models%2Fmnist_saved_model.tar.gz",
|
|             |  "name":                                                                      |
|             | "mnist_saved_model_example",                                                  |
|             |  "processor":                                                                 |
|             | "tensorflow_cpu"                                                              |
|             | }
```

4.6.13. View service processes

You can run the showworkers(w) command to view the status of the service processes.

```
eascmd w [service_name]
```

Examples:

```
$ eascmd w mnist_saved_model_example
[RequestId]: 4E905404-E617-4BD8-85D6-EC5C6A0D0211
+--------------+---------------+--------------------+----------+---------+-------+--------+
|   INNERIP    |    HOSTIP     |     STARTAT        | RESTARTS | STATUS  | READY | REASON |
+--------------+---------------+--------------------+----------+---------+-------+--------+
| 172.24.5.183 | 192.168.65.121 | 2019-02-21 16:35:58 |       0 | Running | [1/1] |       |
+--------------+---------------+--------------------+----------+---------+-------+--------+
```

4.6.14. Call the prediction service

To call a prediction service, you must use the HTTP URL generated when the prediction service is created. The URLs for local testing and online prediction are generated in the same format. The only difference is that they use different hosts. When you send a request to the URL, you must add a token to the HTTP header of the request. The token is generated when you create an image. The following example shows how to call the prediction service from curl. PMML model prediction:

```
$ curl http://{{The domain of EAS}}/api/predict/pmml_example -H 'Authorization: NGE3MzM4YmRhODZj
MTE2NmZjZjNlNTJlNDgyNzM3YzdjMmIwOD****==' -d '[{}]'
[{"prediction_score":0.0902926730924553}]
```

The processor defines the formats of the input and output of the prediction service.

4.6.15. Use Java or C++ to develop a model service

4.6.15.1. What are processors

Processors are program packages that contain the prediction logic. User requests are processed by processors and then returned to clients. A processor contains the logic for loading models and making predictions upon user requests. Machine Learning Platform for AI supports general processors: PMML and TensorFlow. If you want to customize the prediction logic, you must follow the processor development standards to develop a processor.

You can develop a processor in C, C++, and Java languages without the need to use an SDK. You only need to define the relevant classes and functions to develop a processor. This makes it easy for you to debug the processor offline.

4.6.15.2. C/C++ processor

If you use a C or C++ processor, you must define functions Load() and Process(). Function Load() is used to load the model during the initialization process. Function Process() is used to process service calls and return the processing results to clients. The two functions are declared as follows:

```
void *initialize(const char *model_entry, const char *model_config, int *state)
```

Parameter	Type	Description
model_entry	Input parameter	Specifies a model file. This parameter corresponds to the model_entry parameter in the configuration file when you deploy a model service. You can specify a file name or directory, for example, randomforest.pmml or ./model.
model_config	Input parameter	Specifies the custom model configurations. This parameter corresponds to the model_config parameter in the configuration file when you deploy a model service.
state	Output parameter	Indicates whether the model is loaded. Value 0 indicates that the model has been loaded.
N/A	Returned value	The momery address of the user-defined model variable. The value can be any data type.

```
int process(void *model_buf, const void *input_data, int input_size,
        void **output_data, int *output_size)
```

Parameter	Type	Description
model_buf	Input parameter	Specifies the memory address of the model returned by function initialize().
input_data	Input parameter	Specifies the input data. String data and binary data are supported.
input_size	Output parameter	Specifies the size of the input data.
output_data	Output parameter	The data returned by the processor. You must allocate heap memory to the processor. The model will automatically release the memory.
output_size	Output parameter	The size of the data returned by the processor.
N/A	Returned value	If value 0 or 200 is returned, the model service is successfully called. This parameter can return HTTP error codes to clients. If the processor cannot recognize the HTTP error code, it automatically converts it to HTTP error code 400.

Examples

In the following example, no model data is loaded. The prediction service directy returns the request to the client.

```
#include <stdio.h>
#include <string.h>
extern "C" {
    void *initialize(const char *model_entry, const char *model_config, int *state)
    {
        *state = 0;
        return NULL;
    }
    int process(void *model_buf, const void *input_data, int input_size,
            void **output_data, int *output_size)
    {
        if (inputSize == 0) {
            const char *errmsg = "input data should not be empty";
            *outputData = strndup(errmsg, strlen(errmsg));
            *outputSize = strlen(errmsg);
            return 400;
        }
        *outputData = strndup((char *)inputData, inputSize);
        *outputSize = inputSize;
        return 200;
    }
}
```

The processor does not load any model data. It directly returns the input data to the client. You can use Makefile to compile the processor into a .so file.

```
CC=g++
CCFLAGS=-I./ -D_GNU_SOURCE -Wall -g -fPIC
LDFLAGS= -shared -Wl,-rpath=. /

OBJS=processor.o
TARGET=libpredictor.so

all: $(TARGET)

$(TARGET): $(OBJS)
    $(CC) -o $(TARGET) $(OBJS) $(LDFLAGS) -L./

%.o: %.cc
    $(CC) $(CCFLAGS) -c $< -o $@

clean:
    rm -f $(TARGET) $(OBJS)
```

If the processor is reliant on other .so files, package these files with the processor.so file, and then deploy the package.

4.6.15.3. Java processor

The Java processor also uses functions Load() and Process() besides constructors. You only need to define one class for the Java processor. The class is defined as follows:

```
package com.alibaba.eas;

import java.util.*;

public class TestProcessor {
  public TestProcessor(String modelEntry, String modelConfig) {
    /* Passes a model file name to the class and initializes the model. */
  }

  public void Load() {
    /* Loads the model information based on the specified model file name */
  }

  public String Process(String input) {
    /* Preprocesses the input data and outputs the processing results. Currently, only string type data c
an be input and output */
  }

  public static void main(String[] args) {
    /* The main function is optional. It can be used for debugging on your local host. */
  }
}
```

If an exception occurs, the system automatically captures the exception, and returns an error message to the client. An HTTP 400 status code is also returned to the client. You can also customize the logic for capturing exceptions and return an error message to your client as follows:

```
try{
} catch (com.alibaba.fastjson.JSONException e) {
  throw new RuntimeException("bad json format, " + e.getMessage());
}
```

4.7. AutoML (must be activated separately)

4.7.1. Automatic parameter tuning with AutoML

This topic describes the automatic parameter tuning feature of AutoML.

Parameter

1. Log in to machine learning console. In the left-side navigation pane, click **Experiments**.

2. Click an experiment to go to the canvas of the experiment.

   > **Note** This topic uses air quality prediction as an example.

3. In the upper-left corner of the canvas, choose **Auto ML > Auto Parameter Tuning**.
4. On the **Auto Parameter Tuning** page, select an algorithm for parameter tuning, and click **Next**.

   > **Note** You can select only one algorithm to tune at a time.

5. In the **Configure Parameter Tuning** module, set the **Parameter Tuning Method** parameter and click **Next**.

   Alibaba Cloud Machine Learning Platform for AI provides four **parameter tuning methods**. For more information, see Parameter adjustment method.

6. In the **Configure Model Output** module, set the model output parameters and click **Next**.

Parameter	Description
Evaluation Criteria	You can select one evaluation standard from the following four dimensions: **AUC**, **F1-score**, **PRECISION**, and **RECALL**.
Saved Models	You can save up to five models. The system ranks models based on the **Evaluation Criteria** setting you select and save the top ranked models according to the number entered in the **Saved Models** field.
Pass Down Model	This switch is turned on by default. If the switch is turned off, the model generated by the default parameters of the current component are passed down to the node of the subsequent component. If the switch is turned on, the optimal model generated by automatic parameter tuning are passed down to the node of the subsequent component.

7. In the upper-left corner of the canvas, click **Run** to run the automatic parameter tuning algorithm, as shown in the following figure.

   > **Note** After the preceding configuration is complete, the Auto ML switch of the related algorithm is turned on. You can turn the switch on or off as needed.

8. Right-click a model component and choose **Edit AutoML Parameters** from the shortcut menu to modify its AutoML configuration parameters.

Output model display

1. During parameter tuning, right-click the target model component and choose **Parameter Tuning Details** from the shortcut menu.
2. On the **AutoML-Parameter Tuning Details** page, click the **Metrics** tab to view the current

tuning progress and the running status of each model.

3. You can sort candidate models according to indicators (**AUC**, **F1-score**, **Accuracy**, and **Recall Rate**).
4. In the **View Details** column, you can click **Log** or **Parameter** to view the logs and parameters of each candidate model.

Parameter tuning effect display

1. On the **AutoML-Parameter Tuning Details** page, you can click the **Charts** tab to view the **Model Evaluation and Comparison** and **Hyperparameter Iteration Result Comparison** charts.
2. You can view the growth trend of the evaluation indicators of updated parameters in the **Hyperparameter Iteration Result Comparison** chart.

Model storage

1. Log in to machine learning console. In the left-side navigation pane, click **Models**.
2. Click **Experiment Model** to open the experiment model folder.
3. Click the corresponding experiment folder to view the model saved with Auto ML.
4. (Optional) You can apply a model to other experiments by dragging the model to the canvas of the target experiment.

4.7.2. Parameter tuning methods

AutoML supports four parameter tuning methods.

Evolutionary Optimizer

Principle:

1. Randomly selects A parameter candidate sets (where A indicates the **number of exploration samples**).
2. Takes the N parameter candidate sets with higher evaluation indicators as the parameter candidate sets of the next iteration.
3. Continues the exploration within R times (where R indicates the **convergence coefficient**) as the standard deviation range around these parameters to explore new parameter sets. The new parameter sets replace the last A-N parameter sets by the evaluation indicator in the previous round.
4. Iterates the exploration for M rounds (where M indicates the **number of explorations**) until the optimal parameter set is found, according to the preceding logic.

Based on the preceding principle, the final number of models is A + (A - N) × M.

> **Note** The first value of N is A/2 - 1. During iteration, the default value is A/2 - 1 (rounded up).

Parameter	Description

Parameter	Description
Data Splitting Ratio	Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is used for model training and 30% for evaluation.
Exploration Samples	The number of parameter sets of each iteration. The higher the number, the greater the accuracy, the larger the calculation. This parameter must be set to a positive integer in the range of 5 to 30.
Explorations	The number of iterations. The higher the number of iterations, the greater the search accuracy, the larger the calculation. This parameter must be set to a positive integer in the range of 1 to 10.
Convergence Coefficient	Tunes the exploration ranges (R times the standard deviation range search). The smaller the range, the faster the convergence (however, optimal parameters may be missed). Valid values: 0.1 to 1 (one floating point after the decimal point).

Note You must enter the tuning range for each parameter. If the current parameter range is not configured, the parameter range is set by default.

Random Search

Principle:

1. Randomly selects a value for each parameter within the parameter range.
2. Enters random values into a set of parameters for model training.
3. Performs M rounds (where M indicates the number of iterations) and then sorts the output models.

Parameter	Description
Data Splitting Ratio	Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is used for model training and 30% for evaluation.
Iterations	The number of searches in the configured range. Valid values: 2 to 50.

Note You must enter the tuning range for each parameter. If the current parameter range is not configured, the parameter range is set by default.

Grid Search

Principle:

1. Splits the value range of each parameter into N segments (where N indicates the number of split grids).
2. Randomly takes a value from the N segments. Assuming that there are M parameters, N^M parameter sets can be combined.
3. According to the N^M parameter sets, N^M models are generated by training. The models are then sorted.

Parameter	Description
Data Splitting Ratio	Splits input data sources into training and evaluation sets. 0.7 indicates that 70% of the data is used for model training and 30% for evaluation.
Grids	The number of split grids. Valid values: 2 to 10.

Note You must enter the tuning range for each parameter. If the current parameter range is not configured, the parameter range is set by default.

Custom Parameters

- You can enumerate parameter candidate sets. The system then helps score all the combinations of the candidate sets.
- You can define enumeration ranges and separate parameters with commas (,). If the ranges are not configured, the default ranges of parameters are tuned.

4.8. Terms and acronyms

4.8.1. Terms

This topic lists the basic terms used in machine learning.

experiment

A user-created data mining workflow.

project

The basic object in MaxCompute. A project is also known as a workspace. A project contains other objects, such as tables and instances.

component

The minimum operating unit that you can invoke and execute on Apsara Stack Machine Learning Platform for AI. You can use components to import and export data, process data, analyze data, train models, and make predictions.

4.8.2. Acronyms

This topic describes the acronyms used in the Machine Learning Platform for AI User Guide.

MaxCompute

MaxCompute (formerly known as ODPS) is a data processing platform developed by Alibaba Cloud for large-scale data warehousing. MaxCompute can store and compute structured data in batches to meet the requirements of most big data modeling and analysis scenarios.

MaxCompute source and target tables

Tables are data storage objects in MaxCompute. Similar to relational database tables, tables in MaxCompute have a two-dimensional logical structure. A source table is the input of an algorithm node, while a target table is the output of an algorithm node.

5.E-MapReduce (EMR)

5.1. What is E-MapReduce?

E-MapReduce is a managed cluster platform that simplifies running big data frameworks, such as Hadoop, Spark, Kafka, and Storm. E-MapReduce provides you with one-stop big data processing and analysis services, such as managing clusters, jobs, and data.

E-MapReduce is a service that is based on ZStack and uses open-source Apache Hadoop and Spark to process and analyze vast amounts of data. You can use components, such as Apache Hive, Apache Pig, and HBase, in the Hadoop and Spark ecosystems to process and analyze data. You can also use E-MapReduce to import and export data from Alibaba Cloud data stores and databases, such as OSS and ApsaraDB for RDS.

5.2. Introduction

5.2.1. Instructions

Before you use the E-MapReduce service, familiarize yourself with information such as the software configuration of the EMR cluster and the procedure for deploying the service.

5.2.2. Introduction

5.2.2.1. Software configuration

You need to install an operating system and big data components on each ECS instance in an E-MapReduce cluster.

5.2.2.2. Software environment

Software requirements lists the software requirements.

Software requirements

Software	Description
Operating system	CentOS 7 64-bit kernel-3.10.0-693.2.2.el7.x86_64
JDK	OpenJDK 1.8.0

5.2.2.3. Supported components

This topic lists the components of EMR and the supported versions.

Components

Component	Version
Hadoop	2.8.5

Component	Version
Hive	3.1.1
Tez	0.9.1
Spark	2.4.3
Oozie	5.1.0
Hue	4.4.0
Zeppelin	0.8.1
Sqoop	1.4.7
ZooKeeper	3.5.5
Kafka	1.1.1
HBase	1.4.9
Phoenix	4.14.1
Presto	0.22.1
Flink	1.7.2
Ranger	1.2.0
Flume	1.8.0

5.2.2.4. Introduction to components

This section introduces the big data components that can run on an EMR cluster.

Component	Description
Hadoop	• YARN: supports task scheduling and cluster resource management. • HDFS: a distributed file system.
Hive	A Hadoop-based offline data processing system that provides an SQL-like interface to query data. Hive uses tables to store and manage data.
Spark	An in-memory distributed computing framework that supports offline and real-time computing, SQL statements, and machine learning.

Component	Description
Oozie	A job scheduler that supports workflow orchestration by building a directed acyclic graph (DAG). It supports multiple types of jobs.
Hue	A visualized platform used to manage open source components, such as Hadoop, Hive, Oozie, and HBase.
Sqoop	A tool designed to transfer data between HDFS and relational databases.
ZooKeeper	An open source distributed application coordination service. It is an open source implementation of Chubby provided by Google and is an important component of Hadoop and HBase. ZooKeeper solves the consistency problem of distributed applications. Its services include configuration maintenance, domain name services, distributed synchronization, and group services.
Kafka	A distributed messaging system that features high throughput, scalability, reliability, and performance. It is used in real-time computing, log processing, and data aggregation.
HBase	An open source, distributed, and column-oriented database. It is a component of the Apache Hadoop project. Different from typical relational databases, HBase is designed to store unstructured data. HBase is column-oriented rather than row-oriented.
Phoenix	Provides SQL-like statements for you to analyze HBase data.
Presto	A distributed SQL query engine that retrieves large datasets from one or more data sources.

5.2.3. Introduction

5.2.3.1. Deployment

This topic describes available deployment modes for a cluster and supported cluster services.

5.2.3.2. Deployment modes

This topic describes the available deployment modes for an E-MapReduce cluster.

E-MapReduce supports the following deployment modes:

- Hybrid

E-MapReduce supports full-cluster hybrid deployment mode, which means that all components can be deployed in one cluster. Each node in the cluster can provide more than one service.

- Independent

  Only one service is deployed on each E-MapReduce cluster.

5.2.3.3. Supported services

This topic describes services that you can deploy on an EMR cluster.

For more information about services supported by EMR, see List of services.

List of services

Service	Component	Deployment
Hadoop HDFS	NameNode	Deployed on a master node. In a high-availability (HA) cluster, NameNode is deployed on two master nodes.
	DataNode	Deployed on a core node.
	ZKFC	Deployed on a master node. In an HA cluster, ZKFC is deployed on two master nodes.
	JournalNode	• In a non-HA cluster, JournalNode is deployed on a master node, first core node, and second core node. • In an HA cluster, JournalNode is deployed on two master nodes and the first-created core node.
	KMS	Deployed on a master node. Only supports single-node deployment.
	HttpFS	Deployed on a master node. Only supports single-node deployment.
Hadoop YARN	ResourceManager	Deployed on a master node. In an HA cluster, NameNode is deployed on two master nodes.
	NodeManager	Deployed on core nodes.
	JobHistory	Deployed on a master node. Only supports single-node deployment.

Service	Component	Deployment
	TimeLineServer	Deployed on a master node. Only supports single-node deployment.
	WebAppProxyServer	Deployed on a master node. Only supports single-node deployment.
Hive	HiveServer	Deployed on a master node. In an HA cluster, HiveServer is deployed on two master nodes.
	HiveMetaStore	Deployed on the master node. In an HA cluster, HiveMetaStore is deployed on two master nodes.
Spark	JobHistory	Deployed on a master node. Only supports single-node deployment.
Ganglia	GMond	Deployed on all nodes to collect information.
	GMetad	Deployed on a master node. Only supports single-node deployment.
HBase	HMaster	Deployed on a master node. In an HA cluster, HMaster is deployed on two master nodes.
	HRegionServer	Deployed on core nodes.
	ThriftServer	Deployed on a master node. Only supports single-node deployment.
ZooKeeper	ZooKeeper	• In a non-HA cluster, JournalNode is deployed on a master node, first core node, and second core node. • In an HA cluster, JournalNode is deployed on two master nodes and the first core node.
Hue	Hue	Deployed on a master node. In an HA cluster, Hue is deployed on two master nodes.

Service	Component	Deployment
Oozie	Oozie	Deployed on a master node. In an HA cluster, Oozie is deployed on two master nodes.
HAS	HASServer	Deployed on the master node. In an HA cluster, HASServer is deployed on two master nodes.
Knox	Knox	Deployed on the master node. In an HA cluster, Knox is deployed on two master nodes.

5.3. Log on to the E-MapReduce console

This topic describes how to log on to the E-MapReduce console.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

Context

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products** > **Big Data** > **E-MapReduce**.
5. Set Organization and **Region** and click **EMR** to go to the E-MapReduce console.

5.4. Cluster planning and configuration

5.4.1. Cluster planning

5.4.1.1. Gateway clusters

Gateway clusters can serve as a separate job submission node for Hadoop clusters. When you create a gateway cluster, you must associate it with an existing cluster. This facilitates operations on the associated cluster.

A gateway cluster is an independent cluster that consists of multiple instances with the same configurations. Clients, such as Hadoop (HDFS+YARN), Hive, Spark, and Sqoop clients, are deployed on the cluster.

If no gateway cluster is created, jobs of a Hadoop cluster are submitted on the master or core instance of the Hadoop cluster, which consumes the resources of this cluster. After a gateway cluster is created, you can use it to submit jobs of the cluster associated with this gateway cluster. This way, the jobs do not occupy the resources of the associated cluster, and the stability of the master and core instances, especially the master instance, in the associated cluster is improved.

Each gateway cluster can have an independent configuration environment. For example, you can create multiple gateway clusters for one cluster that is shared by multiple departments to meet their business requirements.

5.4.1.2. Disaster recovery in E-MapReduce clusters

This article will introduce disaster recovery of data and dervices in E-MapReduce clusters

Data

HDFS stores the data of each file in blocks, with each block holding multiple copies (three by default). HDFS also makes sure that these copies are stored in different frameworks. In most situations, HDFS stores the first copy in the local framework, the second in the same framework as the first but in different nodes, and the last copy in a different framework.

HDFS scans the data copies regularly. If it finds that a data copy has been lost, HDFS makes another to make sure the number of copies is stable. If a node that stores a copy has been lost, HDFS makes another node to recover the data in that node. In Alibaba Cloud, if you use cloud disks, each cloud disk has three data copies in the back-end. If any of them has an issue, the copies exchange and recover data to ensure reliability.

HDFS is a highly reliable file storage system that can store massive amounts of data. Based on the features of Alibaba Cloud, HDFS can also make backups of the data stored in OSS, providing even greater data reliability.

Services

The core components of HDFS guarantee high availability by making sure that there are at least two nodes to back each other up, such as YARN, HDFS, HiveServer, or Hive Meta. In this way, whenever a node experiences an issue, the nodes can exchange and recover data to ensure that services are not impacted.

5.4.2. Configure clusters

5.4.2.1. Create a cluster

This topic describes how to create an E-MapReduce (EMR) cluster.

Go to the cluster creation page

1. Log on to the E-MapReduce console.
2. Click **Cluster Wizard**.

Configure cluster information

Configure software, hardware, and basic parameters as guided by the wizard.

> **Notice** After a cluster is created, you cannot modify its parameters except for the cluster name. Make sure that all parameters are correct when you create a cluster.

1. Configure software parameters.

Parameter	Description
EMR Version	The major version of EMR. The major version of EMR is a complete open source software environment. It can be regularly updated based on updates made to the internal component software. The major version of EMR is also updated with Hadoop-related software. Clusters of an earlier version cannot be automatically updated to a later version.
Cluster Type	The type of the cluster you want to create. EMR supports the following types of clusters: ○ **Hadoop**: Hadoop clusters provide multiple ecosystem components, such as Hadoop, Hive, Spark, Spark Streaming, Flink, Storm, Presto, Impala, Oozie, and Pig. Hadoop, Hive, and Spark are semi-hosted services and are used to store and compute large-scale distributed data offline. Spark Streaming, Flink, and Storm provide stream computing. Presto and Impala are used for interactive queries. For more information about these components, see the Services section of the Status tab on the Clusters and Services page. ○ **Kafka**: Kafka clusters serve as a semi-hosted, distributed message system with high throughput and scalability. Kafka clusters provide a comprehensive service monitoring system that maintains cluster stability. Kafka clusters are professional, reliable, and secure. You do not need to deploy or maintain these clusters. These clusters are used in scenarios such as log collection and monitoring data aggregation. They can also be used for offline data processing, stream computing, and real-time data analysis. ○ **Druid**: Druid clusters provide a semi-hosted, real-time, and interactive analytic service. These clusters can query big data within milliseconds and ingest data in multiple ways. You can use Druid clusters with services such as EMR Hadoop, EMR Spark, Object Storage Service (OSS), and ApsaraDB for RDS to build a flexible and stable system for real-time queries.

Parameter	Description
Required Services	The default components required for a specific cluster type. After a cluster is created, you can add, start, or stop services on the cluster management page.
Optional Services	The other components that you can specify as required. The relevant service processes for the components you specify are started by default. **Note** The more components you specify, the higher instance specifications a cluster requires to handle the components. You must select the instance type that matches the number of components you specified when you configure the hardware. Otherwise, the cluster may have insufficient resources to run the components.
Advanced Settings	○ **Kerberos Mode**: specifies whether to enable Kerberos authentication for the cluster. This mode is disabled by default because it is not required by clusters created for common users. ○ **Custom Software Settings**: customizes software settings. You can use a JSON file to customize the parameters of basic components required for a cluster, such as Hadoop, Spark, and Hive.

2. Configure hardware parameters.

Section	Parameter	Description
Network Settings	**Zone**	The zone where a cluster is created. Zones are different geographical areas located in the same region. They are interconnected by an internal network. The instance types and disks that are available for you to create a cluster depend on the selected zone. Networks of zones are isolated from each other. If your business requires a low-latency network, we recommend that you deploy instances in the same zone. However, if the zone has insufficient instances, a cluster may fail to be created.
	Network Type	The network type of a cluster. The Virtual Private Cloud (VPC) network type is selected by default.
	VPC	The VPC selected in the region where your cluster resides.
	VSwitch	The VSwitch selected in the zone where your cluster and VPC reside. If no VSwitch is available in the zone, create one.

Section	Parameter	Description
	Security Group Name	The security group to which you want to add your cluster. Only the security groups that you create in EMR are available in the drop-down list. To create a security group, you can directly enter a name in the Security Group Name field. The name must be 2 to 64 characters in length and can contain only letters, digits, underscores (_), and hyphens (-). It must start with a letter.
High Availability	High Availability	Specifies whether to enable the high availability mode. By default, this mode is disabled. In this case, the cluster has only one master node. If the high availability mode is enabled, two master nodes are created in a Hadoop cluster to ensure the availability of the ResourceManager and NameNode processes. HBase clusters support high availability. In the original high availability mode, an HBase cluster uses a core node as the backup of the master node. If you enable the high availability mode described here, an HBase cluster uses two master nodes to achieve high availability, which is more secure and reliable.

Section	Parameter	Description
Instance	Configuration	◦ Node type: ▪ **Master Instance**: runs control processes, such as ResourceManager and NameNode. ▪ **Core Instance**: stores all data of the cluster. You can add core nodes as needed after the cluster is created. ▪ **Task Instance**: improves the computing capabilities of the cluster and stores no data. No task node is configured by default. You can add task nodes as required. ◦ Instance type: Select an instance type for each node type as needed. ◦ Data disk type: Data disk types include standard SSD and ultra disk. Data disk types that are available for you to create the cluster vary with the selected zone and instance type. By default, data disks are released after the cluster is released. For a compute node with local disks, the data disk type is selected by default and cannot be changed. ◦ Data disk size: Valid values: 40 to 32768. Unit: GB. If you use a local disk, the size of the local disk is set by the system and cannot be modified. ◦ Node quantity: You can set the number of nodes of each type as needed. A cluster with the high availability mode disabled must contain three nodes at least. If the high availability mode is enabled, another master node must be added to the cluster.

3. Configure basic parameters.

Section	Parameter	Description
Basic Information	**Cluster Name**	The name of the cluster. The name must be 1 to 64 characters in length and can contain only letters, digits, hyphens (-), and underscores (_).
	Assign Public Network IP	This feature is disabled by default.

Section	Parameter	Description
	Password	The password used to log on to the master node. The password must be 8 to 30 characters in length and contain letters, digits, and special characters. The following special characters are supported: ! @ # $ % ^ & *
Advanced Settings	**Permission Settings**	The RAM roles that allow applications running in a cluster to access other Alibaba Cloud services. You can use the default RAM roles. ○ **EMR Role**: The value is fixed to AliyunEMRDefaultRole and cannot be changed. This RAM role authorizes a cluster to access other Alibaba Cloud services, such as ECS and OSS. ○ **ECS Role**: You can also assign an application role to a cluster. Then, EMR applies for a temporary AccessKey pair when applications running on the compute nodes of that cluster access other Alibaba Cloud services, such as OSS. This way, you do not need to manually enter an AccessKey pair. You can grant the application role access permissions on specific Alibaba Cloud services as required.
	Bootstrap Actions	Optional. You can configure bootstrap actions to run custom scripts before a cluster starts Hadoop.

Confirm the creation

Verify that the configuration is correct and click **Create**.

5.4.2.2. Cluster list and details

This topic describes the parameters in the cluster list, parameters on the cluster details page, and supported cluster operations.

Cluster list

The Cluster Management tab displays the basic information about all of your clusters and the operations supported by each cluster.

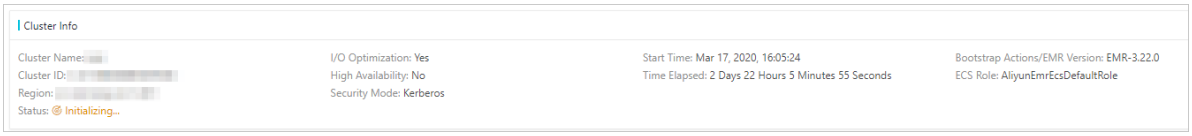
Parameter	Description
Cluster ID/Name	The ID and name of the cluster. Move your pointer over a cluster name and click the icon to modify the cluster name.
Cluster Type	The type of the cluster. Hadoop, Kafka, and Druid cluster types are supported.

Parameter	Description
Status	The status of the cluster. If a cluster experiences an exception, such as a creation failure, error information appears in this column. You can move the pointer over the Help icon next to the error information to view detailed error information.
Created At	The time when the cluster was created.
Time Elapsed	The duration from the point of creation to the current time. After the cluster is released, the time is no longer measured.
Actions	The operations that can be performed on the cluster. The supported operations include: • **Manage**: Go to the Clusters and Services page. • **Details**: View detailed information about the cluster on the Cluster Overview page.

Cluster details

The Cluster Overview page displays detailed information about the cluster and consists of four sections: Cluster Info, Software Info, Network Info, and Instance Info.

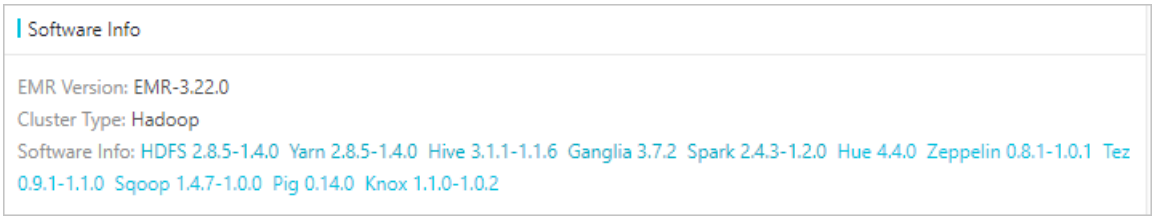
- Cluster Info



Parameter	Description
Cluster Name	The name of the cluster.
Cluster ID	The ID of the cluster.
Region	The region where the cluster resides.
Status	The status of the cluster.
I/O Optimization	Specifies whether I/O optimization is enabled.
High Availability	Specifies whether high availability is enabled.
Security Mode	The security mode. The software of the cluster starts in Kerberos security mode.
Start Time	The time when the cluster was created.
Time Elapsed	The duration from the point of creation to the current time.
Bootstrap Actions/EMR Version	Information about custom scripts and software configuration.

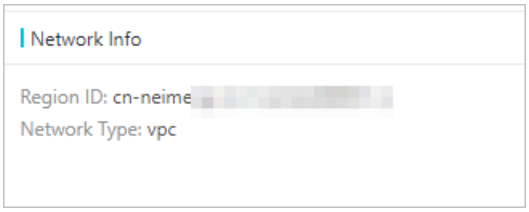
Parameter	Description
ECS Role	The ECS application role. You can assign an application role to a cluster. This way, EMR applies for a temporary AccessKey pair when applications running on the compute nodes of the cluster access other Alibaba Cloud services, such as OSS. You do not need to manually enter an AccessKey pair. You can grant the application role access permissions on specific Alibaba Cloud services as required.

- Software Info



Parameter	Description
EMR Version	The major version of EMR.
Cluster Type	The type of the cluster.
Software Info	All user-installed applications and their versions.

- Network Info



Parameter	Description
Region ID	The zone where the cluster resides, such as cn-hangzhou-b. The value is also the ID of the zone where the ECS instance resides.
Network Type	The network type of the cluster.

- Instance Info

Master instance group information

Parameter	Description
Instances	The current number of instances, and the number of instances that you have requested. Theoretically, the two values are the same. However, the current number is less than the requested number during the cluster creation process.
CPU	The number of CPU cores of a single instance.
Memory	The memory capacity of a single instance.
Data Disk Type	The data disk type and data disk capacity of a single instance.
ECS instances	The information about ECS instances in a master instance group. ○ **ECS ID**: the ID of the ECS instance. ○ **Deployment Status**: the deployment status of the ECS instance. Valid values: Initializing, Normal, and Scaling out. ○ **Public Network IP**: the public IP address of the ECS instance. ○ **Internal Network IP**: the internal IP address of the ECS instance, which can be accessed by all instances in the cluster. ○ **Created At**: the creation time of the ECS instance.

Core instance group information

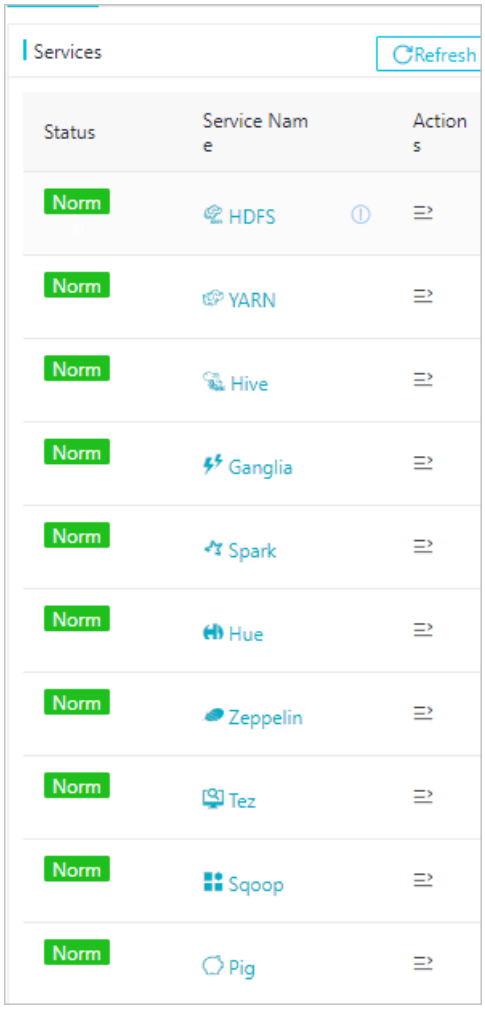
Parameter	Description
Instances	The current number of instances, and the number of instances that you have requested.
CPU	The number of CPU cores of a single instance.
Memory	The memory capacity of a single instance.
Data Disk Type	The data disk type and data disk capacity of a single instance.
ECS instances	The information about ECS instances in a master instance group. ○ **ECS ID**: the ID of the ECS instance. ○ **Deployment Status**: the deployment status of the ECS instance. Valid values: Initializing, Normal, and Scaling out. ○ **Public Network IP**: the public IP address of the ECS instance. ○ **Internal Network IP**: the internal IP address of the ECS instance, which can be accessed by all instances in the cluster. ○ **Created At**: the creation time of the ECS instance.

5.4.2.3. View the running status of services

This topic describes how to view the running status of the services deployed in a cluster, such as HDFS and YARN, on the Clusters and Services page.

1. Log on to the E-MapReduce console.
2. Find the target cluster and click the **cluster ID**.
3. In the Services section of the **Clusters and Services** page, view the status of services.

   The following figure shows the list of services.



   If a service, such as Storm, is not selected when you create the cluster, it is not displayed in the Services section.

   You can click the name of a service to view its details, including the running status, deployment topology, configurations, and historical configuration modifications. A service may be in the **Normal** or **Error** state.

5.4.2.4. Create a gateway cluster

A gateway cluster is composed of ECS instances that reside in the same internal network as an EMR cluster. You can use a gateway cluster to achieve load balancing and security isolation and to submit jobs to the EMR cluster.

Create a gateway cluster in the EMR console

A gateway cluster can be associated only with an EMR Hadoop cluster. Before you create a gateway cluster, make sure that you have created an EMR Hadoop cluster. To create a gateway cluster, perform the following steps:

1. Log on to the E-MapReduce console.
2. In the upper-right corner of the page that appears, click **Create Gateway**.
3. On the **Create Gateway** page, specify the required parameters.
   - **Cluster Name**: the name of the gateway cluster. The name must be 1 to 64 characters in length and can contain only letters, digits, hyphens (-), and underscores (_).
   - **Assign Public Network IP**: specifies whether the gateway cluster is assigned an elastic IP address.
   - **Password**: the password used to log on to the gateway cluster. The password must be 8 to 30 characters in length and must contain uppercase letters, lowercase letters, digits, and special characters.

     The following special characters are supported:

     ! @ # $ % ^ & *
   - **Billing Method**: the billing method of the gateway cluster. The default value is **Pay-As-You-Go**. The system charges you for the gateway cluster on an hourly basis.
   - **Associated Cluster**: the EMR cluster associated with the gateway cluster. The gateway cluster submits jobs to this EMR cluster. The gateway cluster is automatically configured with the same Hadoop environment as the EMR cluster.
   - **Zone**: the zone where the associated cluster resides.
   - **Network Type**: the network type of the associated cluster.
   - **VPC**: the VPC to which the associated cluster belongs.
   - **VSwitch**: the VSwitch you want the gateway cluster to use. Select the VSwitch that corresponds to the zone and VPC.
   - **Security Group Name**: the name of the security group to which the associated cluster belongs.
   - **Gateway Instance**: the available ECS instance types in the current region.
   - **System Disk Type**: the type of the system disks you want the gateway cluster to use. System disk types include standard SSDs and ultra disks. By default, the system disks are released after the relevant cluster is released.
   - **Disk Size**: the size of each system disk. Unit: GiB. Valid values: 40 to 500. Default value: 300.
   - **Data Disk Type**: the type of the data disks you want the gateway cluster to use. Data disk types include standard SSDs and ultra disks. By default, the data disks are released after the relevant cluster is released.
   - **Disk Size**: the size of each data disk. Unit: GiB. Valid values: 200 to 4000. Default value: 300.
   - **Count**: the number of data disks. Valid values: 1 to 10.
4. Click **Create** to save the configurations.

   The created gateway cluster appears in the cluster list. The value of **Status** is **Idle**.

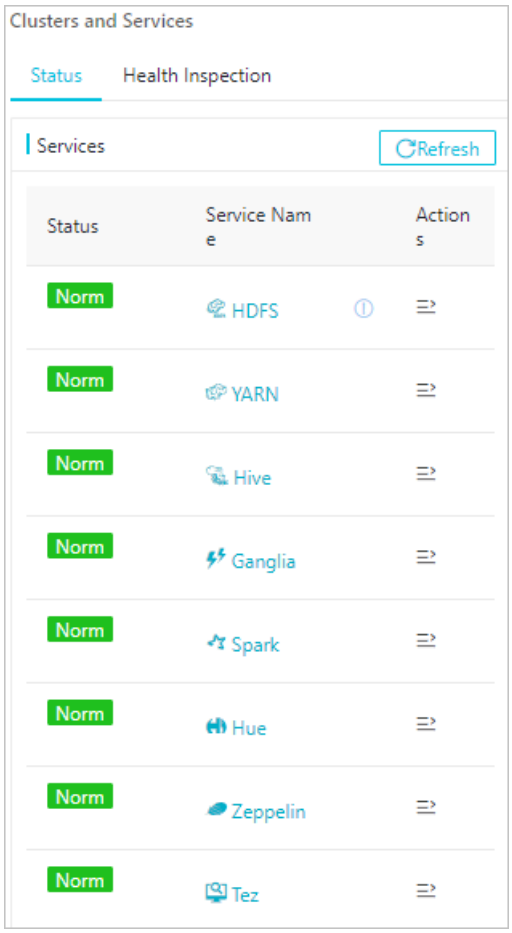
5.4.3. Third-party software

5.4.3.1. Configure parameters for components

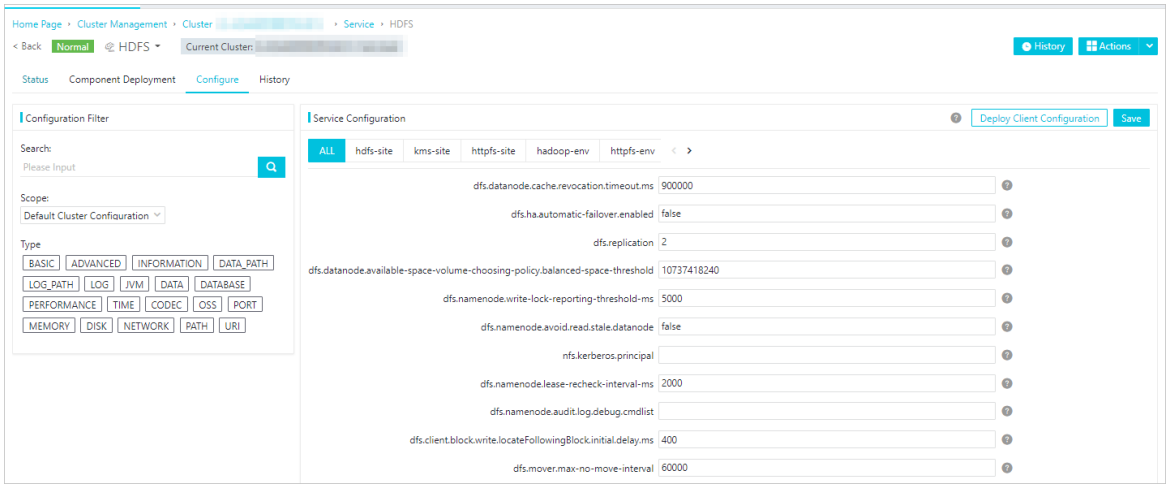
EMR allows you to configure parameters and change parameter settings for components such as HDFS, YARN, Spark, Kafka, and Druid.

Change the parameter settings of an existing component

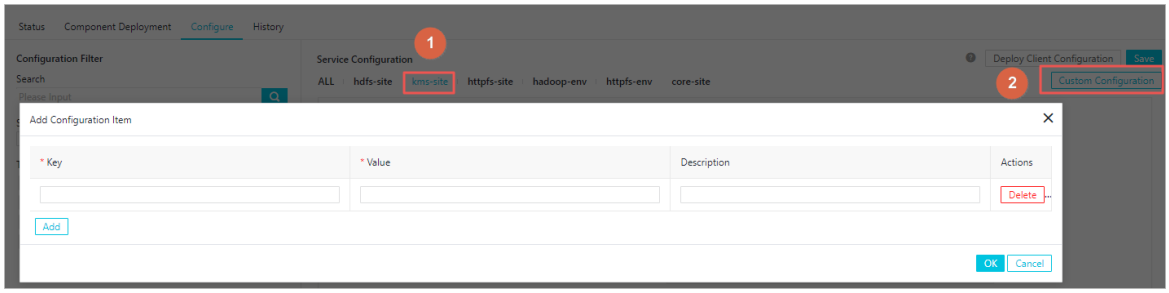
1. Log on to the E-MapReduce console.
2. Find the target cluster and click its ID.
3. In the Services section, click the component for which you want to change a parameter setting, such as **HDFS**.



4. On the HDFS page, click the **Configure** tab.



5. In the **Configuration Filter** section, enter the parameter whose setting you want to change in the search box and click the search icon. You can also specify Scope and Type for the search.
6. In the Service Configuration section, find the target parameter and change its setting. If the parameter does not exist, click the tab that corresponds to the component. Then, in the upper-right corner of the tab, click **Custom Configuration**. In the dialog box that appears, specify the required parameters and click OK.



   - Key: the name of the parameter
   - Value: the value of the parameter
   - Description: the description of the parameter
7. In the upper-right corner of the Service Configuration section, click **Save**. If you want to synchronize the new setting to the gateway cluster associated with the current cluster, select the gateway cluster in the Synchronize to Associated Gateway Cluster section of the **Confirm Changes** dialog box.

Then, make the setting take effect by following the instructions provided in Make configurations take effect.

Make configurations take effect

- Client configuration
  i. After you change the setting of a client configuration parameter, click **Deploy Client Configuration** in the upper-right corner of the Service Configuration section. The **Cluster Activities** dialog box appears.
  ii. Set Target Nodes to Matching Nodes or Designated Node.
  iii. Enter the reason for the change in the Description field.

iv. Click **OK**.

You can click **History** in the upper-right corner of the HDFS page to view the execution status and progress.

- Server configuration

> **Note** After you change the setting of a server configuration parameter, you must restart the current component.

i. In the upper-right corner of the Service Configuration section, click **Deploy Client Configuration**. The **Cluster Activities** dialog box appears.

ii. Specify Target Nodes.

iii. Enter the reason for the change in the Description field.

iv. Click **OK**.

You can click **History** in the upper-right corner of the HDFS page to view the execution status and progress.

Roll back parameter settings

To roll back a parameter setting, perform the following steps:

1. Log on to the E-MapReduce console.
2. Find the target cluster and click its ID.
3. In the Services section, click the component for which you want to change a parameter setting, such as **HDFS**.
4. On the HDFS page, click the **History** tab.
5. Find the target parameter and click **Roll Back** in the Actions column.

Then, make the parameter setting take effect by following the instructions provided in Make configurations take effect.

6.DataHub

6.1. What is DataHub?

DataHub collects, stores, and processes streaming data, allowing you to analyze streaming data and build applications based on the streaming data.

DataHub is a platform designed to process streaming data. You can publish and subscribe to streaming data in DataHub and distribute the data to other platforms. DataHub allows you to analyze streaming data and build applications based on the streaming data.

DataHub collects, stores, and processes streaming data from mobile devices, applications, website services, and sensors. You can use your own applications or Apsara Stack Realtime Compute to process streaming data in DataHub, such as real-time website access logs, application logs, and events. The processing results such as alerts and statistics presented in graphs and tables are updated in real time.

Based on the Apsara system of Alibaba Cloud, DataHub features high availability, low latency, high scalability, and high throughput. DataHub is seamlessly integrated with Realtime Compute, allowing you to use SQL to analyze streaming data.

DataHub can also distribute streaming data to Apsara Stack services such as MaxCompute and Object Storage Service (OSS).

DataHub supports the following features:

- **Data queue**: DataHub automatically generates a cursor for each record in a shard, which can be considered as a logical data queue. The cursor is a unique sequence of numbers. You can improve the performance of a topic by increasing the number of shards in the topic.
- **Offset-based data consumption**: DataHub saves consumption offsets for applications. You can resume data consumption from a saved consumption offset when your application fails.
- **Data synchronization**: Data in DataHub can be automatically synchronized to other Apsara Stack services, including MaxCompute, OSS, AnalyticDB, ApsaraDB RDS for MySQL, Tablestore, and Elasticsearch.
- **Scalable topics**: DataHub allows you to scale in or out topics by splitting or merging shards.

6.2. Usage notes

Before you use DataHub, get familiar with the limits on specific features.

The following table describes the limits of DataHub.

Limits

Item	Limit	Description
Active shards per topic	(0,256]	Each topic can contain up to 256 active shards.
Shards	(0,512]	You can create up to 512 shards in each topic.
HTTP request body size	Up to 4 MB	The size of the HTTP request body cannot exceed 4 MB.

Item	Limit	Description
String size	Up to 1 MB	The size of a string cannot exceed 1 MB.
Merge and split operations on new shards	5s	You cannot merge a shard with another shard or split the shard within the 5s after the shard is created.
Queries per second (QPS)	Up to 5,000	The write QPS limit for each shard is 5,000. Multiple queries in one batch are considered as one query.
Throughput	Up to 5 MB/s	Each shard provides a throughput of up to 5 MB/s.
Projects	Up to 100	You can create up to 100 projects with each account.
Topics per project	Up to 1,000	You can create up to 1,000 topics in each project. Contact the administrator if you need to create more topics.
Time-to-live (TTL) of records	[1,7]	The TTL of each record in a topic ranges from one to seven days.

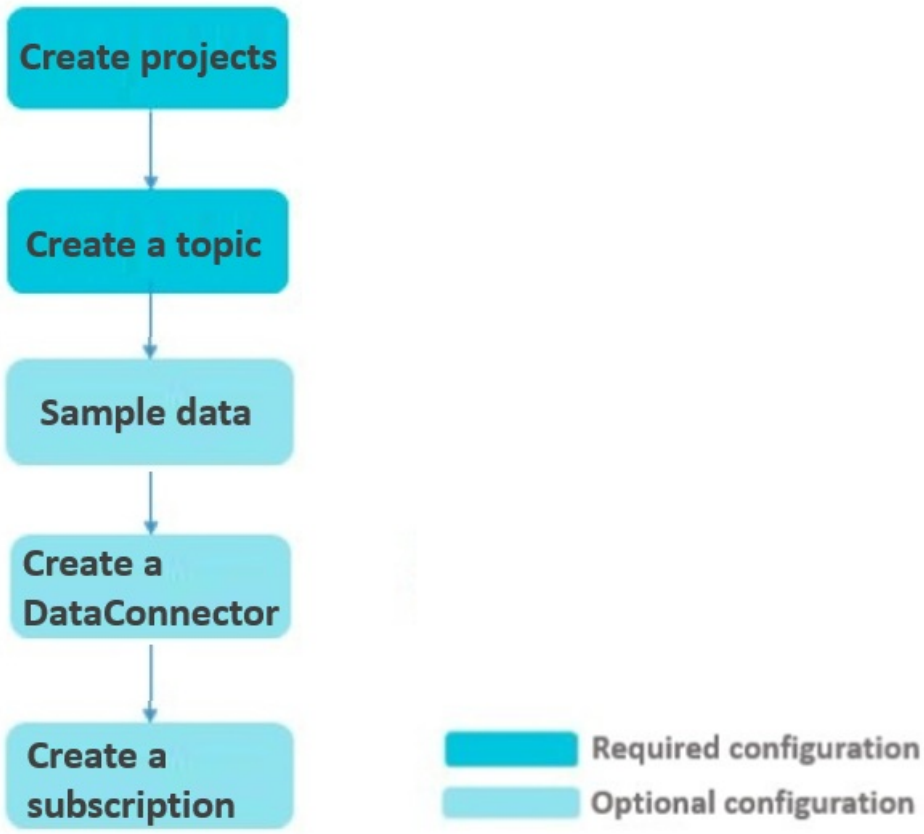
6.3. Quick Start

6.3.1. Overview

This topic describes the procedure of using DataHub.

Procedure shows the procedure of using DataHub.

Procedure



1. Create projects.

   A project is an organizational unit in DataHub and contains one or more topics. When you use DataHub, you must create a project first.

2. Create a topic.

   A topic is the smallest unit for data subscription and publication. You can use topics to distinguish different types of streaming data.

3. Optional. Sample data.

   DataHub supports data sampling. You can sample data of a specific shard.

4. Optional. Create a DataConnector.

   You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data.

5. Optional. Create a subscription.

   The subscription feature of DataHub supports saving consumption offsets on the server and allows applications to resume data consumption from saved consumption offsets. In addition, DataHub supports resetting offsets to ensure that data can be consumed at least once.

6.3.2. Log on to the DataHub console

This topic describes how to log on to the DataHub console by using Google Chrome.

Prerequisites

Before you log on to the DataHub console, make sure that the following prerequisites are met:

- The IP address or domain name of the Apsara Stack console is obtained from deployment engineers.

  The URL used to access the Apsara Stack console is in the following format: https://IP address or domain name of the Apsara Stack console.
- Google Chrome is upgraded to version 42.0.0 or later.

Procedure

1. In the address bar, enter the URL used to log on to the Apsara Stack console and press Enter.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   > **Note** When you log on to the Apsara Stack console for the first time, you must change the password as instructed. For security concerns, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two types of the following characters: letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).
3. Click **Login** to go to the Apsara Stack console homepage.
4. In the top navigation bar, choose **Products** > **Big Data** > **DataHub** to go to the DataHub console. The **Overview** page appears.

6.3.3. Create a project

A project is an organizational unit in DataHub and contains one or more topics. When you use DataHub, you must create a project first. This topic describes how to create a project in the DataHub console.

Prerequisites

An Apsara Stack tenant account is created.

Background information

- DataHub projects are independent from MaxCompute projects. Projects you created in MaxCompute cannot be used in DataHub.
- You can create up to 100 projects with each account.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, click **Create Project**. On the Create Project page, set parameters in the Region and Basic Settings sections and click **Submit**.

6.3.4. Create a topic

A topic is the smallest unit for data subscription and publication. You can use topics to distinguish different types of streaming data. This topic describes how to create a topic in the DataHub console.

Prerequisites

A project is created.

Background information

You can create up to 1,000 topics in each project. Contact the administrator if you need to create more topics.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click **View** in the Operate column.
3. On the project details page, click **Create Topic**.
4. In the Create Topic dialog box, set relevant parameters and click **Create**.

> **Note** DataHub allows you to directly create a topic or create a topic by importing a table schema from MaxCompute.

6.3.5. Sample data

DataHub supports data sampling. You can sample data of a specific shard. This topic describes how to sample data in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Background information

Before you sample data, you must specify the start time and the maximum number of records that you want to sample.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the Shard List tab of the topic details page, find the target shard and click **Sample** in the Operate column.
4. In the dialog box that appears, specify the start time and the maximum number of records that you want to sample and click **Sample**. DataHub samples the records that are written to the shard after the specified time and displays the sampled records in the table below

Sample.

6.3.6. Create a DataConnector

You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data. This topic describes how to create a DataConnector in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Background information

You can synchronize data from DataHub to MaxCompute, AnalyticDB, ApsaraDB RDS for MySQL, Tablestore, OSS, and Elasticsearch.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Connector**. In the Create connector dialog box, select the data warehouse to which data is synchronized.
4. In the Create connector dialog box, set relevant parameters and click **Create**.

6.3.7. Create a subscription

The subscription feature of DataHub supports saving consumption offsets on the server and allows applications to resume data consumption from saved consumption offsets. In addition, DataHub supports resetting offsets to ensure that data can be consumed at least once. This topic describes how to create a subscription in the DataHub console.

Prerequisites

A project and a topic are created and data is written to the topic.

Procedure

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Subscription**. In the Create subscription dialog box, set relevant parameters and click **Create**.

6.4. Access Control

6.4.1. Overview

DataHub allows you to improve data security by granting different permissions to Apsara Stack tenant accounts and RAM user accounts.

DataHub uses Resource Access Management (RAM) for access control. Only users that have been granted the required permissions can access the resources in your department. By default, users do not have permission to access resources in your department. This topic describes how access control for DataHub is achieved by using RAM.

> **Note** An Apsara Stack tenant account is owned by a department and requires no authorization. A RAM user account must be granted permissions by the tenant account.

6.4.2. DataHub resources in RAM

The DataHub resources in RAM are project, topic, and subscription. Subscription is the action that you specify an application to read and process the records in DataHub. DataHub supports RAM authorization of project, topic, and subscription. RAM authorization is not supported at the shard level.

In RAM, each resource type has a general description and each specific object of the resource type has a description. For example, the description of a project that resides in a certain region is *acs:dhs:$region:$accountid:projects/$projectName*. *$region, $accountid, and $projectName* indicate the region that the project resides, the user ID, and the project name.

Resource description

Resource type	Description
SingleProject	acs:dhs:$region:$accountid:projects/$projectName
AllProject	acs:dhs:$region:$accountid:projects/*
SingleTopic	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName
AllTopic	acs:dhs:$region:$accountid:projects/$projectName/topics/*
SingleSubscription	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName/subscriptions/$subId
AllSubscription	acs:dhs:$region:$accountid:projects/$projectName/topics/$topicName/subscriptions/*

6.4.3. API

DataHub provides application programming interfaces (APIs) for projects, topics, shards, subscriptions, and records. Before you can call the API operations, you must grant corresponding permissions to the RAM user by using RAM authorization policies.

The RAM authorization policy and resource type for each API operation is described as follows:

API operations for projects

API operations for projects

Operation name	RAM authorization policy	Resource type
CreateProject	dhs:CreateProject	AllProject
ListProject	dhs:ListProject	AllProject
DeleteProject	dhs:DeleteProject	SingleProject
GetProject	dhs:GetProject	SingleProject
UpdateProject	dhs: UpdateProject	SingleProject

API operations for topics

API operations for topics

Operation name	RAM authorization policy	Resource type
CreateTopic	dhs:CreateTopic	AllTopic
ListTopic	dhs:ListTopic	AllTopic
DeleteTopic	dhs:DeleteTopic	SingleTopic
GetTopic	dhs:GetTopic	SingleTopic
UpdateTopic	dhs: UpdateTopic	SingleTopic

API operations for subscriptions

API operations for subscriptions

Operation name	RAM authorization policy	Resource type
CreateSubscription	dhs:CreateSubscription	AllSubscription
ListSubscription	dhs:ListSubscription	AllSubscription
DeleteSubscription	dhs:DeleteSubscription	SingleSubscription
GetSubscription	dhs:GetSubscription	SingleSubscription
UpdateSubscription	dhs: UpdateSubscription	SingleSubscription
CommitOffset	dhs:CommitOffset	SingleSubscription
GetOffset	dhs:GetOffset	SingleSubscription

API operations for shards

API operations for shards

Operation name	RAM authorization policy	Resource type
ListShard	dhs:ListShard	SingleTopic
MergeShard	dhs:MergeShard	SingleTopic
SplitShard	dhs:SplitShard	SingleTopic

API operations for shards

API operations for shards

Operation name	RAM authorization policy	Resource type
PutRecords	dhs:PutRecords	SingleTopic
GetRecords	dhs:GetRecords	SingleTopic
GetCursor	dhs:GetRecords	SingleTopic

6.4.4. Conditions

This section describes conditions that can be applied to the RAM authorization policies for DataHub.

Conditions that can be applied to the RAM authorization policies for DataHub are as follows:

RAM authorization policy conditions for DataHub

Condition keyword	Description	Valid value
acs:SourceIp	The IP address range that can access the specified object.	Any valid IP address. Wildcard masks are supported.
acs:SecureTransport	Indicates whether HTTPS is used to access the specified object.	true/false
acs:MFAPresent	Indicates whether the specified object can be accessed by multiple clients.	true/false
acs:CurrentTime	The time that the specified object can be accessed.	This keyword must be described in ISO 8601 format.

6.4.5. Sample RAM authorization policy content

6.4.5.1. AliyunDataHubFullAccess

This section describes how to set the AliyunDataHubFullAccess policy content.

The authorization policy content can be set as follows:

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": "dhs:*",
   "Resource": "*",
   "Effect": "Allow"
  }
 ]
}
```

6.4.5.2. AliyunDataHubReadOnlyAccess

This section describes how to set the AliyunDataHubReadOnlyAccess policy content.

The authorization policy content can be set as follows:

```
{
 "Version": "1",
 "Statement": [
  {
   "Action": ["dhs:List*", "dhs:Get*"],
   "Resource": "*",
   "Effect": "Allow"
  }
 ]
}
```

6.5. Data Acquisition

6.5.1. Overview

In addition to SDK and local file uploads, DataHub supports various data acquisition tools to help you quickly collect data to DataHub.

This section describes how to acquire data by using Fluentd, Logstash, and Oracle GoldenGate (OGG).

6.5.2. Fluentd

This topic describes how to install and use the DataHub plug-in for Fluentd.

Developed based on the open-source data collector Fluentd, the DataHub plug-in for Fluentd is easy to install and is used to write the collected data to DataHub.

Install the DataHub plug-in for Fluentd

- Install the plug-in by using RubyGems

```
gem install fluent-plugin-datahub
```

> **Notice** We recommend that you change the gem source to https://ruby.taobao.org/.

- Install the plug-in by using a local installation package

  The agent must be installed in Linux. Before you install the agent, install Ruby. For users who have not installed Fluentd, a full installation package for installing both Fluentd and DataHub plug-in is provided. For users who have installed Fluentd, an installation package of the DataHub plug-in is provided.

  - If you have not installed Fluentd, download the full installation package and run the following commands to install Fluentd with the DataHub plug-in:

    > **Notice** Fluentd 0.12.23 is provided in the full installation package.

    ```
    $ tar -xzvf fluentd-with-datahub-0.12.23.tar.gz
    $ cd fluentd-with-dataHub
    $ sudo sh install.sh
    ```

  - If you have installed Fluentd, download the installation package of the DataHub plug-in for Fluentd and run the following command to install the plug-in.

    ```
    $ sudo gem install --local fluent-plugin-dataHub-0.0.2.gem
    ```

Use cases

Case 1: Collect CSV files

This example shows how to write the incremental content of a Comma-Separated Values (CSV) file to DataHub in quasi-real time by using the DataHub plug-in for Fluentd. The following CSV file is used in this example:

```
0,qe614c760fuk8judu01tn5x055rpt1,true,100.1,14321111111
1,znv1py74o8ynn87k66o32ao4x875wi,true,100.1,14321111111
2,7nm0mtpgo1q0ubuljjjx9b000ybltl,true,100.1,14321111111
3,10t0n6pvonnan16279w848ukko5f6l,true,100.1,14321111111
4,0ub584kw88s6dczd0mta7itmta10jo,true,100.1,14321111111
5,1ltfpf0jt7fhvf0oy4lo8m3z62c940,true,100.1,14321111111
6,zpqsfxqy9379lmcehd7q8kftntrozb,true,100.1,14321111111
7,ce1ga9aln346xcj761c3iytshyzuxg,true,100.1,14321111111
8,k5j2id9a0ko90cykl40s6ojq6gruyi,true,100.1,14321111111
9,ns2zcx9bdip5y0aqd1tdicf7bkdmsm,true,100.1,14321111111
10,54rs9cm1xau2fk66pzyz62tf9tsse4,true,100.1,14321111111
```

Each line is a record to be written to DataHub. Columns are separated by commas (,). Save the CSV file as /temp/test.csv on the local computer. The following table shows the schema of the DataHub topic to which the CSV file is written.

DataHub topic schema

Field	Data type
id	BIGINT
name	STRING
gender	BOOLEAN
salary	DOUBLE
my_time	TIMESTAMP

After you edit the Fluentd configuration file based on the CSV file and topic schema, run the following command to start Fluentd to write the CSV file to DataHub:

```
${FLUENTD_HOME}/fluentd-with-dataHub/bin/fluentd -c fluentd_test.conf
```

Use the following Fluentd configuration file in this example:

```
<source>
  @type tail
  path /xxx/yyy (Specify the path of the CSV file.)
  tag test1
  format csv
  keys id,name,gender,salary,my_time
</source>

<match test1>
  @type dataHub
  access_id your_app_id
  access_key your_app_key
  endpoint http://ip:port
  project_name test_project
  topic_name fluentd_performance_test_1
  column_names ["id", "name", "gender", "salary", "my_time"]
  flush_interval 1s
  buffer_chunk_limit 3m
  buffer_queue_limit 128
  dirty_data_continue true
  dirty_data_file /xxx/yyy (Specify the path of the dirty record file.)
  retry_times 3
  put_data_batch_size 1000
</match>
```

Case 2: Collect Log4j logs

The following format of Log4j logs is used in this example:

```
11:48:43.439 [qtp1847995714-17] INFO  AuditInterceptor - [c2un5sh7cu52ek6am1ui1m5h] end /web/v1/p
roject/tefe4mfurtix9kwwyrvfqd0m/node/0m0169kapshvgc3ujskwkk8g/health GET, 4061 ms
```

Use the following Fluentd configuration file in this example:

```
<source>
  @type tail
  path bayes.log
  tag test
  format /(? <request_time>\d\d:\d\d:\d\d.\d+)\s+\[(? <thread_id>[\w\-]+)\]\s+(? <log_level>\w+)\s+(
? <class>\w+)\s+-\s+\[(? <request_id>\w+)\]\s+(? <detail>.+)/
</source>

<match test>
  @type dataHub
  access_id your_access_id
  access_key your_access_key
  endpoint http://ip:port
  project_name test_project
  topic_name dataHub_fluentd_out_1
  column_names ["thread_id", "log_level", "class"]
</match>
```

Parameter description

Input configuration

Parameter	Description
tag test1	The tag, which is mapped to the destination information by using the specified regular expression.
format csv	The format of the file from which data is collected.
keys id,name,gender,salary,my_time	The columns to be collected from the CSV file. The column names must be the same as those in the schema of the destination DataHub topic.

Output configuration

Parameter	Description
shard_id 0	The ID of the shard to which all records are written. By default, all records are written to the shard by polling. The default ID is 0.
shard_keys ["id"]	The column used as the shard key. Hashed shard key values are used as indexes for writing data.
flush_interval 1	The interval between data flushes. The default value is 60s.

Parameter	Description
buffer_chunk_limit 3m	The maximum size of a chunk. Unit: k or m, which indicates KB or MB. We recommend you set the maximum size to 3 MB.
buffer_queue_limit 128	The maximum length of the chunk queue. Both the buffer_chunk_limit and buffer_queue_limit parameters determine the size of the buffer. The default value is 128 MB.
put_data_batch_size 1000	The number of records to be written to DataHub at a time. In this example, 1,000 records are written to DataHub each time.
retry_times 3	The number of retries for writing data to DataHub. Default value: 3.
retry_interval 3	The retry interval at which data is written. Unit: seconds. Default value: 3.
dirty_data_continue true	Specifies whether to ignore dirty records. The value true indicates that the plug-in retries the operation for a specified number of times before it writes the dirty records to the dirty record file.
dirty_data_file /xxx/yyy	The directory where the dirty record file is stored.
column_names ["id"]	The name of the columns to be written to DataHub.

6.5.3. Logstash

This topic describes how to install and use Logstash to import data to DataHub and export data from DataHub.

Logstash is a distributed log collection framework. It is often used with Elasticsearch and Kibana, known as the ELK Stack, for log data analysis. To support a wider variety of data inputs, DataHub offers Output and Input plug-ins for data transfer with Logstash. By using Logstash, you can access more than 30 types of data sources in the Logstash open source community, such as files, Syslog logs, Redis logs, Log4j logs, Apache logs, and NGINX logs. Logstash also supports filter plug-ins for customizing the fields to be transferred. This topic demonstrates how to use Logstash with DataHub.

Install Logstash and DataHub plug-ins

Java Runtime Environment (JRE) 7 or later is required to run Logstash. If the JRE version does not meet the requirement, several features of Logstash are unavailable. You can install Logstash and DataHub plug-ins with one click by downloading and decompressing the software package or install Logstash and DataHub plug-ins separately.

- Install Logstash and DataHub plug-ins with one click: Download the software package.

  Run the following commands to decompress the package and go to the software directory:

```
$ tar -xzvf logstash-with-datahub-2.3.0.tar.gz
$ cd logstash-with-datahub-2.3.0
```

- Install Logstash and DataHub plug-ins separately
  - Install Logstash. For more information, see the documentation on the official website of Logstash.
  - Install the DataHub Output plug-in for Logstash. You can use this plug-in to import data to DataHub.
  - Install the DataHub Input plug-in for Logstash. You can use this plug-in to export data from DataHub.

Use cases

Case 1: Collect Log4j logs

This example shows how to collect unstructured Log4j logs and derive a structure out of the logs by using Logstash. The following format of Log4j logs is used in this example:

```
20:04:30.359 [qtp1453606810-20] INFO  AuditInterceptor - [13pn9kdr5tl84stzkmaa8vmg] end /web/v1/project/fhp4clxfbu0w3ym2n7ee6ynh/statistics? executionName=bayes_poc_test GET, 187 ms
```

In this example, you can derive a structure out of the logs and transfer the data to DataHub. The following table shows the schema of the DataHub topic to which the Log4j logs are written.

DataHub topic schema

Field	Data type
request_time	STRING
thread_id	STRING
log_level	STRING
class_name	STRING
request_id	STRING
detail	STRING

Use the following configuration of the Logstash task in this example:

```
input {
    file {
        path => "${APP_HOME}/log/bayes.log"
        start_position => "beginning"
    }
}

filter{
    grok {
        match => {
          "message" => "(? <request_time>\d\d:\d\d:\d\d\.\d+)\s+\[(? <thread_id>[\w\-]+)\]\s+(? <log_le
vel>\w+)\s+(? <class_name>\w+)\s+\-\s+\[(? <request_id>\w+)\]\s+(? <detail>.+)"
        }
    }
}

output {
    datahub {
        access_id => "Your accessId"
        access_key => "Your accessKey"
        endpoint => "Endpoint"
        project_name => "project"
        topic_name => "topic"
        #shard_id => "0"
        #shard_keys => ["thread_id"]
        dirty_data_continue => true
        dirty_data_file => "/Users/ph0ly/trash/dirty.data"
        dirty_data_file_max_size => 1000
    }
}
```

Case 2: Collect CSV files

This example shows how to use Logstash to collect CSV files. The following CSV file is used in this example:

```
1111,1.23456789012E9,true,14321111111000000,string_dataxxx0,
2222,2.23456789012E9,false,14321111111000000,string_dataxxx1
```

The following table shows the schema of the DataHub topic to which the CSV file is written.

DataHub topic schema

Field	Data type
col1	BIGINT
col2	DOUBLE
col3	BOOLEAN
col4	TIMESTAMP
col5	STRING

Use the following configuration of the Logstash task in this example:

```
input {
  file {
    path => "${APP_HOME}/data.csv"
    start_position => "beginning"
  }
}

filter{
  csv {
    columns => ['col1', 'col2', 'col3', 'col4', 'col5']
  }
}

output {
  datahub {
    access_id => "Your accessId"
    access_key => "Your accessKey"
    endpoint => "Endpoint"
    project_name => "project"
    topic_name => "topic"
    #shard_id => "0"
    #shard_keys => ["thread_id"]
    dirty_data_continue => true
    dirty_data_file => "/Users/ph0ly/trash/dirty.data"
    dirty_data_file_max_size => 1000
  }
}
```

Case 3: Consume data from DataHub

Use the following configuration of the Logstash task in this example:

```
input {
  datahub {
    access_id => "Your accessId"
    access_key => "Your accessKey"
    endpoint => "Endpoint"
    project_name => "test_project"
    topic_name => "test_topic"
    interval=> 5
    #cursor => {
    #   "0"=>"20000000000000000000000003110091"
    #   "2"=>"20000000000000000000000003110091"
    #   "1"=>"20000000000000000000000003110091"
    #   "4"=>"20000000000000000000000003110091"
    #   "3"=>"20000000000000000000000003110000"
    #}
    shard_ids => []
    pos_file => "/home/admin/logstash/logstash-2.3.0/pos_file"
  }
}

output {
  file {
    path => "/home/admin/logstash/logstash-2.3.0/output"
  }
}
```

Start Logstash

Run the following command to start Logstash:

```
$LOGSTASH_HOME/bin/logstash -f <The preceding configuration file> -b 256
```

> **Note** -f is followed by the path of the configuration file. -b is followed by the number of records transferred to DataHub at a time. Default value: 125.

Parameters

The following table describes the parameters of the DataHub Output plug-in.

Parameters of the DataHub Output plug-in

Parameter	Description
access_id	Required. The AccessKey ID of your Apsara Stack tenant account.
access_key	Required. The AccessKey secret of your Apsara Stack tenant account.
endpoint	Required. The endpoint used to access DataHub.
project_name	Required. The name of the DataHub project.
topic_name	Required. The name of the DataHub topic.
retry_times	Optional. The maximum number of retries. The value -1 indicates unlimited retries. The value 0 indicates no retries. A value greater than 0 indicates the specified number of retries. Default value: -1.
retry_interval	Optional. The interval between retries. Unit: seconds. Default value: 5.
shard_keys	Optional. The key of the shard. The hash of the key value is used to map the ID of the shard to which the records are written. If the shard_keys and shard_id parameters are not specified, the system polls the shards to decide which shard the records are written to.
shard_id	Optional. The ID of the shard where records are written. If the shard_keys and shard_id parameters are not specified, the system polls the shards to decide which shard the records are written to.
dirty_data_continue	Optional. Specifies whether to ignore dirty records. The value true indicates that dirty records are to be ignored. Default value: false. If you set the value to true, you must specify the dirty_data_file parameter.
dirty_data_file	Optional. The name of the dirty record file. The dirty record file is divided into .part 1 and .part 2. The most recent records are stored in .part 2.
dirty_data_file_max_size	Optional. The maximum size of the dirty record file. This value is for reference only.

The following table describes the parameters of the DataHub Input plug-in.

Parameters of the DataHub Input plug-in

Parameter	Description
access_id	Required. The AccessKey ID of your Apsara Stack tenant account.
access_key	Required. The AccessKey secret of your Apsara Stack tenant account.
endpoint	Required. The endpoint used to access DataHub.
project_name	Required. The name of the DataHub project.
topic_name	Required. The name of the DataHub topic.

Parameter	Description
retry_times	Optional. The maximum number of retries. The value -1 indicates unlimited retries. The value 0 indicates no retries. A value greater than 0 indicates the specified number of retries. Default value: -1.
retry_interval	Optional. The interval between retries. Unit: seconds. Default value: 5.
shard_ids	Optional. The shards in which records are to be consumed. If this parameter is not specified, records in all the shards are consumed.
cursor	Optional. The sequence number of the record from which the consumption begins. The consumption starts from the earliest record by default.
pos_file	Required. The checkpoint file, which is used to reset the consumption offset.

6.5.4. Oracle GoldenGate

This topic describes how to install and use Oracle GoldenGate (OGG).

OGG is a tool for log-based structured data replication across heterogeneous environments. It is used for data backup between primary and secondary Oracle databases. It is also used to synchronize data from Oracle databases to other databases such as IBM Db2 and MySQL databases. OGG must be deployed in the source and destination databases. It is composed of the following components: Manager, Extract, data pump, Collector, and Replicat.

- Manager is the control process of OGG. A Manager process must be running on the source and destination databases. It is responsible for starting, stopping, and monitoring other processes.
- Extract is a process that captures data from the source database or transaction logs. You can configure the Extract process for initial data loads and incremental data synchronization. For initial data loads, Extract captures a set of data directly from their source objects. To keep source data synchronized to the destination database, Extract captures incremental DML and DDL operations after the initial data loading has taken place. This topic describes incremental data synchronization.
- A data pump is a secondary Extract group within the source OGG configuration. In a typical configuration with a data pump, the primary Extract group writes to a trail on the source database. The data pump reads the trail and sends the DML or DDL operations over the network to a remote trail on the destination database.
- Collector is a process on the destination database, which receives data from the source database and generates trail files.
- Replicat is a process that reads the trail on the destination database, reconstructs the DML or DDL operations, and then applies them to the destination database.

The DataHub agent for OGG offers the Replicat feature that applies the updated data to DataHub by analyzing the trail. The data in DataHub is processed in real time by using Realtime Compute and can be archived into MaxCompute.

The following example shows how to synchronize incremental data from an Oracle database to DataHub and process the data in DataHub.

Install OGG

Prerequisites:

- You have installed the Oracle database client.
- You have obtained the OGG installation package for the source database. We recommend that you use OGG V12.1.2.1.
- You have obtained the OGG Adapters installation package for the destination database. We recommend that you use OGG Application Adapters 12.1.2.1.
- You have installed Java 7.

Follow these steps to install OGG:

1. Install OGG for the source database.
   i. Extract the OGG installation package for the source database and the following directories appear:

```
drwxr-xr-x install
drwxrwxr-x response
-rwxr-xr-x runInstaller
drwxr-xr-x stage
```

   ii. Install dependencies in response/oggcore.rsp. The OGG response file template is as follows:

```
oracle.install.responseFileVersion=/oracle/install/rspfmt_ogginstall_response_schema
#The installation option, which must reflect the installed Oracle version. Specify ORA11g for i
nstalling OGG for Oracle Database 11g.
INSTALL_OPTION=ORA11g
#The location in which OGG is installed.
SOFTWARE_LOCATION=/home/oracle/u01/ggate
#Indicates whether to start the Manager after installation.
START_MANAGER=false
#The port number of the Manager process.
MANAGER_PORT=7839
#The location of the Oracle database.
DATABASE_LOCATION=/home/oracle/u01/app/oracle/product/11.2.0/dbhome_1
#The location that stores the inventory files. This parameter is not required to be configured.
INVENTORY_LOCATION=
#The UNIX group of the inventory directory. In this example, OGG is installed by using the ogg
_test Oracle account. You can also create a dedicated account for OGG as necessary.
UNIX_GROUP_NAME=oinstall
```

iii. Run the following command to install OGG:

```
runInstaller -silent -responseFile {YOUR_OGG_INSTALL_FILE_PATH}/response/oggcore.rsp
```

> **Note**
>
> In this example, OGG is installed in */home/oracle/u01/ggate* and the installation logs are stored in */home/oracle/u01/ggate/cfgtoollogs/oui*. The OGG installation is complete when the following message appears in the silentInstall{time}.log file:
>
> ```
> The installation of Oracle GoldenGate Core was successful.
> ```

iv. Run the following command and enter `CREATE SUBDIRS` as required to create OGG directories:

```
/home/oracle/u01/ggate/ggsci
```

2. Perform Oracle configurations in the source database.

   Navigate to *sqlplus: sqlplus / as sysdba* as the database administrator and complete the following configurations:

```
#Create a tablespace.
create tablespace ATMV datafile '/home/oracle/u01/app/oracle/oradata/uprr/ATMV.dbf' size 100
m autoextend on next 50m maxsize unlimited;

 #Create a user named ogg_test. The password is also set to ogg_test.
create user ogg_test identified by ogg_test default tablespace ATMV;

 #Grant required privileges to ogg_test.
grant connect,resource,dba to ogg_test;

 #Check whether supplemental logging is enabled for the database.
Select SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA
_UI, SUPPLEMENTAL_LOG_DATA_FK, SUPPLEMENTAL_LOG_DATA_ALL from v$database;

 #If the result is NO, enable supplemental logging.
alter database add supplemental log data;
alter database add supplemental log data (primary key, unique,foreign key) columns;
 #Enable rollback.
alter database drop supplemental log data (primary key, unique,foreign key) columns;
alter database drop supplemental log data;

 #Enable all column logging at the database level. Note: Even when all column logging is enabled,
only primary key columns are logged for a delete operation.
```

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
 #Enable the forced logging mode.
alter database force logging;
 #Run the marker_setup.sql script.
@marker_setup.sql
 #Run the ddl_setup.sql script.
@ddl_setup.sql
 #Run the role_setup.sql script.
@role_setup.sql
 #Grant the GGS_GGSUSER_ROLE to ogg_test.
grant GGS_GGSUSER_ROLE to ogg_test;
 #Run the ddl_enable.sql script to enable the DDL trigger.
@ddl_enable.sql
 #Run the ddl_pin script to improve the performance of the DDL trigger.
@ddl_pin ogg_test
 #Run the sequence.sql script.
@sequence.sql
#
alter table sys.seq$ add supplemental log data (primary key) columns;
```

3. Configure the Manager process on the source database.

   Start the Oracle GoldenGate Software Command Interface (GGSCI) and perform the following steps:

   i. Run the following command to configure the Manager process:

   ```
   edit params mgr
   PORT 7839
   DYNAMICPORTLIST  7840-7849
   USERID ogg_test, PASSWORD ogg_test
   PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 7
   LAGREPORTHOURS 1
   LAGINFOMINUTES 30
   LAGCRITICALMINUTES 45
   PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
   PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
   ```

   ii. Run the following command to start the Manager process. The logs are stored in ggate/dirrpt.

   ```
   start mgr
   ```

iii. Run the following command to check whether the Manager process is running:

```
info mgr
```

iv. Run the following command to view the Manager parameter file:

```
view params mgr
```

4. Configure the Extract process on the source database.

   Start the GGSCI and perform the following steps:

   i. Run the following command to configure the Extract process. In the following example, the group name of the process is extract.

```
edit params extractEXTRACT extract
SETENV (NLS_LANG="AMERICAN_AMERICA.AL32UTF8")
DBOPTIONS   ALLOWUNUSEDCOLUMN
USERID ogg_test, PASSWORD ogg_test
REPORTCOUNT EVERY 1 MINUTES, RATE
NUMFILES 5000
DISCARDFILE ./dirrpt/ext_test.dsc, APPEND, MEGABYTES 100
DISCARDROLLOVER AT 2:00
WARNLONGTRANS 2h, CHECKINTERVAL 3m
EXTTRAIL ./dirdat/st, MEGABYTES 200
DYNAMICRESOLUTION
TRANLOGOPTIONS CONVERTUCS2CLOBS
TRANLOGOPTIONS RAWDEVICEOFFSET 0
DDL &
INCLUDE MAPPED OBJTYPE 'table' &
INCLUDE MAPPED OBJTYPE 'index' &
INCLUDE MAPPED OBJTYPE 'SEQUENCE' &
EXCLUDE OPTYPE COMMENT
DDLOPTIONS  NOCROSSRENAME  REPORT
TABLE    OGG_TEST.*;
SEQUENCE  OGG_TEST.*;


GETUPDATEBEFORES
```

   ii. Run the following command to add an Extract process. Replace extract in the following command with your actual group name.

```
add ext extract,tranlog, begin now
```

iii. Run the following command to delete an Extract process. In the following example, the process name is DP_TEST.

```
delete ext DP_TEST
```

iv. Run the following command to create a trail, associate the trail with the Extract group named extract, and set the maximum file size in the trail to 200 megabytes:

```
add exttrail ./dirdat/st,ext extract, megabytes 200
```

v. Run the following command to start the Extract process. The logs are stored in ggate/dirrpt.

```
start extract extract
```

Note After the Extract process configuration is complete, you can view the changes to the database in the files stored in the *ggate/dirdat* directory.

5. Create a DEFGEN parameter file.

i. Start the GGSCI in the source database. In GGSCI, run the following command to create a DEFGEN parameter file and copy the file to the dirdef directory in the destination database:

```
edit params defgen
DEFSFILE ./dirdef/ogg_test.def
USERID ogg_test, PASSWORD ogg_test
table OGG_TEST. *;
```

ii. Run the following command from the shell to create a DEFGEN parameter file named ogg_test.def:

```
./defgen paramfile ./dirprm/defgen.prm
```

6. Install and configure OGG in the destination database.

i. Extract the OGG installation package to the destination database.

ii. Copy the dirdef/ogg_test.def file in the source database to dirdef of the destination database.

iii. Start the GGSCI and run the following command to create the default directories of OGG:

```
create subdirs
```

iv. Run the following command to configure the Manager process:

```
edit params mgr
PORT 7839
DYNAMICPORTLIST  7840-7849
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 7
LAGREPORTHOURS 1
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
PURGEDDLHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
PURGEMARKERHISTORY MINKEEPDAYS 3, MAXKEEPDAYS 7
```

v. Run the following command to start the Manager process:

```
start mgr
```

7. Configure a data pump in the source database.

Start the GGSCI and perform the following steps:

i. Run the following command to configure a data pump:

```
edit params pump
EXTRACT pump
RMTHOST xx.xx.xx.xx, MGRPORT 7839, COMPRESS
PASSTHRU
NUMFILES 5000
RMTTRAIL ./dirdat/st
DYNAMICRESOLUTION
TABLE    OGG_TEST.*;
SEQUENCE  OGG_TEST.*;
```

ii. Run the following command to create a data-pump Extract process. The process reads from the specified trail.

```
add ext pump,exttrailsource ./dirdat/st
```

iii. Run the following command to create a trail and set the maximum file size in the trail to 200 megabytes:

```
add rmttrail ./dirdat/st,ext pump,megabytes 200
```

iv. Run the following command to start the data pump:

```
start pump
```

Note After the data pump is started, you can view the trail files in the dirdat directory of the destination database.

8. Install and configure the DataHub agent for OGG.
   i. Run the following command to configure the *JAVA_HOME and LD_LIBRARY_PATH* environment variables and specify the configurations in the ~/.bash_profile:

```
export JAVA_HOME=/xxx/xxx/jrexx
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$JAVA_HOME/lib/amd64:$JAVA_HOME/lib/amd64/server
```

   ii. After the environment variables are configured, restart the Manager process in the destination database.
   iii. Download the DataHub agent for OGG and extract the installation package.
   iv. Modify the javaue.properties and log4j.properties files in the conf sub-directory of the installation directory and replace {YOUR_HOME} with the target path of the extracted files:

```
gg.handlerlist=ggdatahub
gg.handler.ggdatahub.type=com.aliyun.odps.ogg.handler.datahub.DatahubHandler
gg.handler.ggdatahub.configureFileName={YOUR_HOME}/datahub-ogg-plugin/conf/configure.xml
goldengate.userexit.nochkpt=false
goldengate.userexit.timestamp=utc
gg.classpath={YOUR_HOME}/datahub-ogg-plugin/lib/*
gg.log.level=debug
jvm.bootoptions=-Xmx512m -Dlog4j.configuration=file:{YOUR_HOME}/datahub-ogg-plugin/conf/log4j.properties -Djava.class.path=ggjava/ggjava.jar
```

   v. Modify the configure.xml file in the conf sub-directory of the installation directory as follows:

```
<? xml version="1.0" encoding="UTF-8"? >
<configue>

  <defaultOracleConfigure>
    <! --(Required) The Oracle database system identifier (SID).-->
    <sid>100</sid>
    <! --The schema of the Oracle table, which can be overwritten by oracleSchema in the column mappings. At least one of them must be specified.-->
    <schema>ogg_test</schema>
  </defaultOracleConfigure>

  <defalutDatahubConfigure>
    <! --(Required) The endpoint of DataHub.-->
    <endPoint>YOUR_DATAHUB_ENDPOINT</endPoint>
    <! --The DataHub project, which can be overwritten by datahubProject in the column map
```

```
pings. At least one of them must be specified.-->
        <project>YOUR_DATAHUB_PROJECT</project>
        <! --The AccessKey ID for accessing DataHub, which can be overwritten by datahubAcces
sId in the column mappings. At least one of them must be specified.-->
        <accessId>YOUR_DATAHUB_ACCESS_ID</accessId>
        <! --The AccessKey Secret for accessing DataHub, which can be overwritten by datahubA
ccessKey in the column mappings. At least one of them must be specified.-->
        <accessKey>YOUR_DATAHUB_ACCESS_KEY</accessKey>
        <! --The column in DataHub that indicates the data update type, which can be overwritte
n by ctypeColumn in the column mappings.-->
        <ctypeColumn>optype</ctypeColumn>
        <! -- The column in DataHub that indicates the data update time, which can be overwritte
n by ctimeColumn in the column mappings.-->
        <ctimeColumn>readtime</ctimeColumn>
        <! -- The column in DataHub that indicates the sequence number of the updated data, w
hich can be overwritten by cidColumn in the column mappings. The sequence number increase
s as more data are updated, but may not be consecutive.-->
        <cidColumn>record_id</cidColumn>
    </defalutDatahubConfigure>

    <! --The approach to handling errors. If an error occurs, the system either ignores the error
and continues running or retries the operation repeatedly.-->

    <! --(Optional) The maximum number of records operated at one time. Default value: 1000.-
->
    <batchSize>1000</batchSize>

    <! --(Optional) The format that the timestamp is converted into. Default: yyyy-MM-dd HH:m
m:ss.-->
    <defaultDateFormat>yyyy-MM-dd HH:mm:ss</defaultDateFormat>

    <! --(Optional) Indicates whether the system needs to ignore dirty records. Default value: f
alse.-->
    <dirtyDataContinue>true</dirtyDataContinue>

    <! --(Optional) The dirty record file name. Default value: datahub_ogg_plugin.dirty-->
    <dirtyDataFile>datahub_ogg_plugin.dirty</dirtyDataFile>

    <! --(Optional) The maximum size of the dirty record file. Unit: MB. Default value: 500.-->
    <dirtyDataFileMaxSize>200</dirtyDataFileMaxSize>
```

```
  <! --(Optional) The maximum number of retries if an error occurs. -1: Unlimited. 0: No retrie
s. n: The number of retries. Default value: -1.-->
  <retryTimes>0</retryTimes>

  <! --(Optional) The interval between retries. Unit: milliseconds. Default value: 3000.-->
  <retryInterval>4000</retryInterval>

  <! --(Optional) The checkpoint file name. Default value: datahub_ogg_plugin.chk.-->
  <checkPointFileName>datahub_ogg_plugin.chk</checkPointFileName>

  <mappings>
    <mapping>
      <! --The schema of the Oracle table.-->
      <oracleSchema></oracleSchema>
      <! --(Required) The Oracle table name.-->
      <oracleTable>t_person</oracleTable>
      <! --The DataHub project name.-->
      <datahubProject></datahubProject>
      <! --The AccessKey ID for accessing DataHub.-->
      <datahubAccessId></datahubAccessId>
      <! --The AccessKey Secret for accessing DataHub.-->
      <datahubAccessKey></datahubAccessKey>
      <! --(Required) The DataHub topic name.-->
      <datahubTopic>t_person</datahubTopic>
      <ctypeColumn></ctypeColumn>
      <ctimeColumn></ctimeColumn>
      <cidColumn></cidColumn>
      <columnMapping>
        <! --
        src: (Required) The column names in the Oracle table.
        dest: (Required) The column names in the DataHub topic.
        destOld: (Optional) The DataHub topic column that records the data before it is upd
ated.
        isShardColumn: (Optional) Indicates whether the shard ID is generated based on th
e hash key value, which can be overwritten by shardId. Default value: false.
        isDateFormat: Indicates whether the timestamp is converted into a string based on
dateFormat. Default value: true. If you set the value to false, the data type in the source dat
abase must be long.
        dateFormat: The format that the timestamp is converted into. If this parameter is lef
t blank, the default format is used.
        -->
```

```
            <column src="id" dest="id" isShardColumn="true"  isDateFormat="false" dateFormat
="yyyy-MM-dd HH:mm:ss"/>
            <column src="name" dest="name" isShardColumn="true"/>
            <column src="age" dest="age"/>
            <column src="address" dest="address"/>
            <column src="comments" dest="comments"/>
            <column src="sex" dest="sex"/>
            <column src="temp" dest="temp" destOld="temp1"/>
        </columnMapping>

        <! --(Optional) The ID of the shard prioritized to be written into.-->
        <shardId>1</shardId>
      </mapping>
   </mappings>
</configue>
```

vi. Run the following command in GGSCI to start the DataHub writer:

```
edit params dhwriter
extract dhwriter
getEnv (JAVA_HOME)
getEnv (LD_LIBRARY_PATH)
getEnv (PATH)
CUSEREXIT ./libggjava_ue.so CUSEREXIT PASSTHRU INCLUDEUPDATEBEFORES, PARAMS "{YOUR_
HOME}/datahub-ogg-plugin/conf/javaue.properties"
sourcedefs ./dirdef/ogg_test.def
table OGG_TEST. *;
```

vii. Run the following command to add a DataHub writer:

```
add extract dhwriter, exttrailsource ./dirdat/st
```

viii. Run the following command to start the writer:

```
start dhwriter
```

Use case

For example, you have an Oracle table that stores order information. The table has three columns. The column names are oid, pid, and num, which indicate order ID, product ID, and product quantity. You can synchronize incremental data to DataHub by using the DataHub agent for OGG. The steps are as follows:

> **Note** Before performing incremental data synchronization, you must synchronize existing data from the source table to MaxCompute by using DataX.

1. Create a topic in DataHub. The schema of the topic is as follows:

```
string record_id, string optype, string readtime, bigint oid_before, bigint oid_after, bigint pid_befo
re, bigint pid_after, bigint num_before, bigint num_after
```

2. Make sure that you have completed the deployment of the DataHub agent for OGG. Then configure the column mappings as follows:

```
<ctypeColumn>optype</ctypeColumn>
        <ctimeColumn>readtime</ctimeColumn>
        <cidColumn>record_id</cidColumn>
        <columnMapping>
          <column src="oid" dest="oid_after" destOld="oid_before" isShardColumn="true"/>
          <column src="pid" dest="pid_after" destOld="pid_before"/>
          <column src="num" dest="num_after" destOld="num_before"/>
        </columnMapping>
```

> **Note** The optype parameter indicates the type of the data update. Valid values of the optype parameter are I, D, and U, which represent an insert, delete, and update operation, respectively. The readtime parameter indicates the time of the data update.

3. When the agent can run properly, data updates are synchronized from the source table to DataHub.

6.6. Data Archive

6.6.1. Overview

You can synchronize real-time data from DataHub to other data warehouses by using DataConnectors so that you can analyze and process historical data.

The following topics describe how to synchronize data from DataHub to MaxCompute.

6.6.2. Archive to MaxCompute

6.6.2.1. Create a DataConnector

This topic describes how to create a DataConnector to synchronize data from DataHub to MaxCompute.

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.
3. On the topic details page, click **Connector**. In the Create connector dialog box, select the data warehouse to which data is synchronized.
4. In the Create connector dialog box, set relevant parameters and click **Create**.

Note

The following table describes the parameters of the DataConnector for synchronizing data from DataHub to MaxCompute.

Parameter	Description
Project Name	The name of the MaxCompute project to which data in the topic is synchronized.
Table Name	The name of the MaxCompute table to which data in the topic is synchronized.
AccessID and AccessKey	The AccessKey pair used to access MaxCompute. The AccessKey pair must belong to a RAM user that has CreateInstance, Desc, and Alter permissions on the MaxCompute table.
Partition Mode	The method used to create partitions. Valid values: SYSTEM_TIME, EVENT_TIME, USER_DEFINE, and META_TIME. If you select SYSTEM_TIME, partitions are created based on the recording time. If you select EVENT_TIME, partitions are created based on the value of the event_time field. When you create the topic, you must define a field named event_time for the topic and set its data type to TIMESTAMP. The value of the event_time field must be accurate to microseconds. If you select USER_DEFINE, partitions are created based on the user-defined partition key.
Partition Config	The format of the time based on which partitions are created. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME.
Time Range	The interval of creating partitions. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME. The minimum value is 15 minutes.
Timezone	The time zone of the time based on which partitions are created. This parameter takes effect only when you set the Partition Mode parameter to SYSTEM_TIME, EVENT_TIME, or META_TIME.
Start Time	The time when data synchronization starts.

6.6.2.2. View data synchronization details

This section describes how to view data synchronization details after a DataConnector is created.

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click **View** in the Operate column.
3. On the project details page, find the target topic and click **View** in the Operate column.
4. On the topic details page, click the **Connector** tab. On the Connector tab, find the target DataConnector and click **View** in the Operate column.

> **Notice** You can restart or stop a DataConnector. Exercise caution when you perform the operations.

6.7. Metric statistics

This topic describes how to view the metric statistics of a topic in DataHub.

In the DataHub console, you can view the metric statistics of topics in quasi-real-time, such as QPS and throughput. The following metrics are available:

- Read and write QPS
- Read and write records per second (RPS)
- Read and write throughput, measured in KB per second
- Read and write latency, measured in microseconds per request

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the target project and click **View** in the Operate column.
3. On the project details page, find the target topic and click **View** in the Operate column.
4. On the topic details page, click the **Metric Statistics** tab.

You can view the metric statistics for a specified time range.

6.8. Data subscription

6.8.1. Overview

Resumable consumption is required in scenarios where you consume data in DataHub topics and want to resume the consumption from the time when your application fails. If you need to resume consumption, you must save the current consumption offset and make sure that the service for saving consumption offsets supports high availability. This increases the complexity of developing applications. The subscription feature of DataHub supports saving consumption offsets to the server to solve the preceding problem. You only need to enable this feature and add a few lines of code to your application to obtain a consumption offset maintenance service with high availability.

In addition, the subscription feature allows you to reset consumption offsets. This ensures that the data can be consumed at least once. For example, if an error occurs when your application processes the data consumed in a specific time period and you need to consume the data again, you can reset the consumption offset without restarting the application. Your application automatically consumes data from the specified consumption offset.

6.8.2. Create a subscription

You can create subscriptions only in the DataHub console. Make sure that your account is authorized to subscribe to topics of the specified project.

Perform the following steps to create a subscription:

1. Log on to the DataHub console.
2. In the left-side navigation pane, click **Project Manager**. On the Project List page, find the

target project and click View in the Operate column. On the project details page, find the target topic and click **View** in the Operate column.

3. On the topic details page, click **Subscription**. In the Create subscription dialog box, set relevant parameters and click **Create**.
4. After the subscription is created, click the **Subscription List** tab on the topic details page to view the subscriptions of the topic.

> **Note** You can click **Reset** or **Delete** in the Operate column of a subscription.
>
> - **Reset**: resets the consumption offset of the subscription to the required time. Specify the time in the *mm-dd-yyyy HH:MM:SS* format.
> - **Delete**: permanently deletes the subscription, including all consumption offsets that are saved for the subscription.

6.8.3. Use case

The subscription feature allows you to save consumption offsets. You can use the read and write capabilities of DataHub with the capability of saving consumption offsets in scenarios where you must save consumption offsets after data is read.

The following sample code is used for reference only.

```
// The following sample code consumes data from a saved consumption offset and submit consumptio
n offsets during consumption.
public void offset_consumption(int maxRetry) {
    String endpoint = "<YourEndPoint>";
    String accessId = "<YourAccessId>";
    String accessKey = "<YourAccessKey>";
    String projectName = "<YourProjectName>";
    String topicName = "<YourTopicName>";
    String subId = "<YourSubId>";
    String shardId = "0";
    List<String> shardIds = Arrays.asList(shardId);
    // Create a DataHub client.
    DatahubClient datahubClient = DatahubClientBuilder.newBuilder()
        .setDatahubConfig(
            new DatahubConfig(endpoint,
                // Specify whether to enable binary data transmission. The server of V2.12 or later sup
ports binary data transmission.
                new AliyunAccount(accessId, accessKey), true))
        .build();
    RecordSchema schema = datahubClient.getTopic(projectName, topicName).getRecordSchema();
    OpenSubscriptionSessionResult openSubscriptionSessionResult = datahubClient.openSubscriptionS
ession(projectName, topicName, subId, shardIds);
    SubscriptionOffset subscriptionOffset = openSubscriptionSessionResult.getOffsets().get(shardId);
```

```
    // 1. Obtain the cursor of the record at the current consumption offset. If the record expired or the r
ecord is not consumed, obtain the cursor of the first record within the TTL of the topic.
    String cursor = "";
    // If the sequence number is smaller than 0, the record is not consumed.
    if (subscriptionOffset.getSequence() < 0) {
        // Obtain the cursor of the first record within the TTL of the topic.
        cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OLDEST).getCurs
or();
    } else {
        // Obtain the cursor of the next record.
        long nextSequence = subscriptionOffset.getSequence() + 1;
        try {
            // If the SeekOutOfRange error is returned after you obtain the cursor based on the sequence
number, the record expired.
            cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SEQUENCE, nex
tSequence).getCursor();
        } catch (SeekOutOfRangeException e) {
            // Obtain the cursor of the first record within the TTL of the topic.
            cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OLDEST).getCu
rsor();
        }
    }
    // 2. Read records and save consumption offsets. In this example, you read tuple records and save c
onsumption offsets each time 1,000 records are read.
    long recordCount = 0L;
    // Read 1,000 records each time.
    int fetchNum = 1000;
    int retryNum = 0;
    int commitNum = 1000;
    while (retryNum < maxRetry) {
        try {
            GetRecordsResult getRecordsResult = datahubClient.getRecords(projectName, topicName, shar
dId, schema, cursor, fetchNum);
            if (getRecordsResult.getRecordCount() <= 0) {
                // If no records can be read, pause the thread for 1s and continue to read records.
                System.out.println("no data, sleep 1 second");
                Thread.sleep(1000);
                continue;
            }
            for (RecordEntry recordEntry : getRecordsResult.getRecords()) {
                // Consume data
```

```
            // Consume data.
            TupleRecordData data = (TupleRecordData) recordEntry.getRecordData();
            System.out.println("field1:" + data.getField("field1") + "\t"
                + "field2:" + data.getField("field2"));
            // Save the consumption offset after the data is consumed.
            recordCount++;
            subscriptionOffset.setSequence(recordEntry.getSequence());
            subscriptionOffset.setTimestamp(recordEntry.getSystemTime());
            // commit offset every 1000 records
            if (recordCount % commitNum == 0) {
              // Submit the consumption offset.
              Map<String, SubscriptionOffset> offsetMap = new HashMap<>();
              offsetMap.put(shardId, subscriptionOffset);
              datahubClient.commitSubscriptionOffset(projectName, topicName, subId, offsetMap);
              System.out.println("commit offset successful");
            }
          }
          cursor = getRecordsResult.getNextCursor();
        } catch (SubscriptionOfflineException | SubscriptionSessionInvalidException e) {
          // The subscription session is exited. The Offline exception indicates that the subscription is of
fline. The SessionChange exception indicates that the subscription is consumed by other clients.
          e.printStackTrace();
          throw e;
        } catch (SubscriptionOffsetResetException e) {
          // The consumption offset is reset. You must obtain the version information of the consumption
offset again.
          SubscriptionOffset offset = datahubClient.getSubscriptionOffset(projectName, topicName, subI
d, shardIds).getOffsets().get(shardId);
          subscriptionOffset.setVersionId(offset.getVersionId());
          // After the consumption offset is reset, you must obtain the cursor of the record at the consu
mption offset again. The method for obtaining the cursor depends on the method of resetting the con
sumption offset.
          // If both the sequence number and timestamp are specified to reset the consumption offset, y
ou can obtain the cursor based on the sequence number or the timestamp.
          // If only the sequence number is specified to reset the consumption offset, you can obtain the
cursor only based on the sequence number.
          // If only the timestamp is specified to reset the consumption offset, you can obtain the cursor
only based on the timestamp.
          // Generally, preferentially obtain the cursor based on the sequence number. If the cursor faile
d to be obtained based on the sequence number or the timestamp, obtain the cursor of the earliest re
cord.
```

```
            cursor = null;
            if (cursor == null) {
                try {
                    long nextSequence = offset.getSequence() + 1;
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SEQUENCE,
nextSequence).getCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by SEQUENCE failed, try to get cursor by SYSTEM_TIME");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.SYSTEM_TI
ME, offset.getTimestamp()).getCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by SYSTEM_TIME failed, try to get cursor by OLDEST");
                }
            }
            if (cursor == null) {
                try {
                    cursor = datahubClient.getCursor(projectName, topicName, shardId, CursorType.OLDEST).g
etCursor();
                    System.out.println("get cursor successful");
                } catch (DatahubClientException exception) {
                    System.out.println("get cursor by OLDEST failed");
                    System.out.println("get cursor failed!!") ;
                    throw e;
                }
            }
        } catch (LimitExceededException e) {
            // limit exceed, retry
            e.printStackTrace();
            retryNum++;
        } catch (DatahubClientException e) {
            // other error, retry
            e.printStackTrace();
            retryNum++;
        } catch (Exception e) {
            e.printStackTrace();
```

```
            System.exit(-1);
        }
    }
}
```

Note

- When you start the application for the first time, your application consumes data from the earliest record. During the running of the application, you can refresh the Subscription List tab in the console.
- If you reset the consumption offset by clicking Reset in the console during the consumption, your application automatically detects the change of the consumption offset and consumes data from the specified consumption offset. When the application catches OffsetResetedException, the application calls the getSubscriptionOffset method to query the latest consumption offset from the server. Then, the application can consume data from the latest consumption offset.
- Note that a shard in a subscription cannot be consumed by multiple threads or processes at the same time. Otherwise, the consumption offset submitted by a thread is overwritten by that submitted by another thread and the server cannot determine to which thread the saved consumption offset belongs. In this case, the server throws OffsetSessionChangedException. We recommend that you exit the subscription session to check whether data is repeatedly consumed if this exception is caught.

6.9. Collaborative consumption

6.9.1. Note

DataHub-client-library encapsulates the Java SDK and integrates the consumer for collaborative consumption and the producer for distributing data evenly among shards.

6.9.2. Overview

Offset-based data consumption

The offset-based data consumption feature allows you to save consumption offsets to the server. A consumption offset consists of the sequence number of a record and the timestamp when the record is written to DataHub.

You can create a subscription for a topic and submit the consumption offset to the server after specific data is consumed. When you application starts the next time, the application can obtain the consumption offset from the server and consume data from the next record. The consumption offsets must be saved on the server so that your application can consume data from a submitted consumption offset after shards are reallocated. This is a prerequisite for collaborative consumption.

You do not need to manually submit consumption offsets in the consumer. You only need to specify the interval of submitting consumption offsets in the configurations of the consumer. The system considers that the previous records are consumed when it reads records. If the interval of submitting consumption offsets is exceeded, the system submits a consumption offset again. If the consumption offset fails to be submitted and your application is interrupted, the consumption offset may fail to be submitted in time. In this case, your application may repeatedly consume specific data.

Collaborative consumption

The collaborative consumption feature automatically allocates shards when multiple consumers consume a topic at the same time. This feature simplifies the data processing of clients.

> **Note** Manual shard allocation is difficult because multiple consumers may reside on different machines. If multiple consumers that subscribe to the same topic are in the same consumer group, a shard can be allocated to only one consumer in the consumer group.

Example:

Assume that A, B, and C are three consumer instances and the topic has 10 shards.

1. When the consumer instance A is started at first, 10 shards are allocated to it.
2. When the other two consumer instances are started, the shards are reallocated in the following way: four to A, three to B, and three to C.
3. When one of the shards consumed by the consumer instance A is split into two and the two shards are released after consumption, the shards are reallocated in the following way: four to A, four to B, and three to C.
4. When the consumer instance C is stopped, the shards are reallocated in the following way: six to A and five to B.

Heartbeat

You must use the heartbeat feature to notify the server of the status of consumer instances. If the server has not received heartbeats from a consumer instance after the specified interval, the server considers that the consumer instance is stopped. When the status of a consumer instance changes, the server reallocates shards. The server returns the new allocation plan in heartbeat requests. Therefore, the client takes time to detect reallocation of shards.

6.9.3. Maven dependencies and JDK

Maven dependencies

```
<dependency>
    <groupId>com.aliyun.datahub</groupId>
    <artifactId>datahub-client-library</artifactId>
    <version>1.0.6-public</version>
</dependency>
```

JDK

```
jdk: >= 1.7
```

6.9.4. Use case

The following sample code is for reference only.

Initialize the producer

```
String endpoint = "http://dh-cn-hangzhou.aliyuncs.com";
String accessId = "<YourAccessKeyId>";
String accessKey = "<YourAccessKeySecret>";
String projectName = "<YourProjectName>";
String topicName = "<YourTopicName>";
ProducerConfig config = new ProducerConfig(endpoint, accessId, accessKey);
Producer producer = new Producer(projectName, topicName, config);
```

Write data to DataHub

```
RecordSchema schema = new RecordSchema();
schema.addField(new Field("field1", FieldType.STRING));
schema.addField(new Field("field2", FieldType.BIGINT));
List<RecordEntry> recordEntries = new ArrayList<>();
for (int cnt = 0; cnt < 10; ++cnt) {
    RecordEntry entry = new RecordEntry();
    entry.addAttribute("key1", "value1");
    entry.addAttribute("key2", "value2");
    TupleRecordData data = new TupleRecordData(schema);
    data.setField("field1", "testValue");
    data.setField("field2", 1);
    entry.setRecordData(data);
    recordEntries.add(entry);
}
int maxRetry = 3;
while (true) {
    try {
        producer.send(records, maxRetry);
        break;
    } catch (MalformedRecordException e) {
        // malformed RecordEntry
    } catch (InvalidParameterException e) {
        // invalid param
    } catch (ResourceNotFoundException e) {
        // project, topic or shard not found, sometimes caused by split/merge shard
    } catch (DatahubClientException e) {
        // network or other exceptions exceeded retry limit
    }
}
// close before exit
producer.close();
```

Initialize the consumer

```
String endpoint = "http://dh-cn-hangzhou.aliyuncs.com";
String accessId = "<YourAccessKeyId>";
String accessKey = "<YourAccessKeySecret>";
String projectName = "<YourProjectName>";
String topicName = "<YourTopicName>";
String SubId = "<YourSubscriptionId>";
// 1. If you need to use the collaborative consumption feature, specify the subscription ID.
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, config);
// 2. If you need to use the offset-based data consumption feature instead of the collaborative consu
mption feature, specify the subscription ID and the shards to be read by the consumer.
List<String> assignment = Arrays.asList("0", "1", "2");
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, assignment, config);
// 3. If you do not need to use the collaborative consumption feature nor the offset-based data consu
mption feature, specify the subscription ID, the shards to be read by the consumer, and the consumpti
on offset.
Map<String, Offset> offsetMap = new HashMap<>();
// If both the sequence number and timestamp are specified but the sequence number is invalid, obtai
n the cursor based on the timestamp.
offsetMap.put("0", new Offset(100, 1548573440756L));
// If only the sequence number is specified, obtain the cursor based on the sequence number.
offsetMap.put("1", new Offset().setSequence(1));
// If only the timestamp is specified, obtain the cursor based on the timestamp.
offsetMap.put("2", new Offset().setTimestamp(1548573440756L));
ConsumerConfig config = new ConsumerConfig(endpoint, accessId, accessKey);
Consumer consumer = new Consumer(projectName, topicName, SubId, offsetMap, config);
```

Read data from DataHub

```
int maxRetry = 3;
boolean stop = false;
while (! stop) {
    try {
        while (true) {
            RecordEntry record = consumer.read(maxRetry);
            if (record ! = null) {
                TupleRecordData data = (TupleRecordData) record.getRecordData();
                System.out.println("field1:" + data.getField(0) + ", field2:" + data.getField("field2"));
            }
        }
    } catch (SubscriptionSessionInvalidException | SubscriptionOffsetResetException e) {
        // subscription exception, will not recover
        // print some log or just use a new consumer
        consumer.close();
        consumer = new Consumer(TEST_PROJECT, TEST_TOPIC, TEST_SUB_ID, config);
    } catch (ResourceNotFoundException | InvalidParameterException e) {
        // - project, topic, shard, subscription not found
        // - seek out of range
        // - sometimes shard operation cause ResourceNotFoundException
        // should make sure if resource exists, print some log or just exit
    } catch (DatahubClientException e) {
        // - network or other exception exceed retry limit
        // can just sleep and retry
    }
}
// close before exit
consumer.close();
```

6.9.5. Usage notes

A consumer or producer cannot access DataHub by using multiple threads. If you need to use multiple threads, specify a different consumer or producer for each thread.

7.Quick BI

7.1. What is Quick BI?

This topic describes the concept and features of Quick BI.

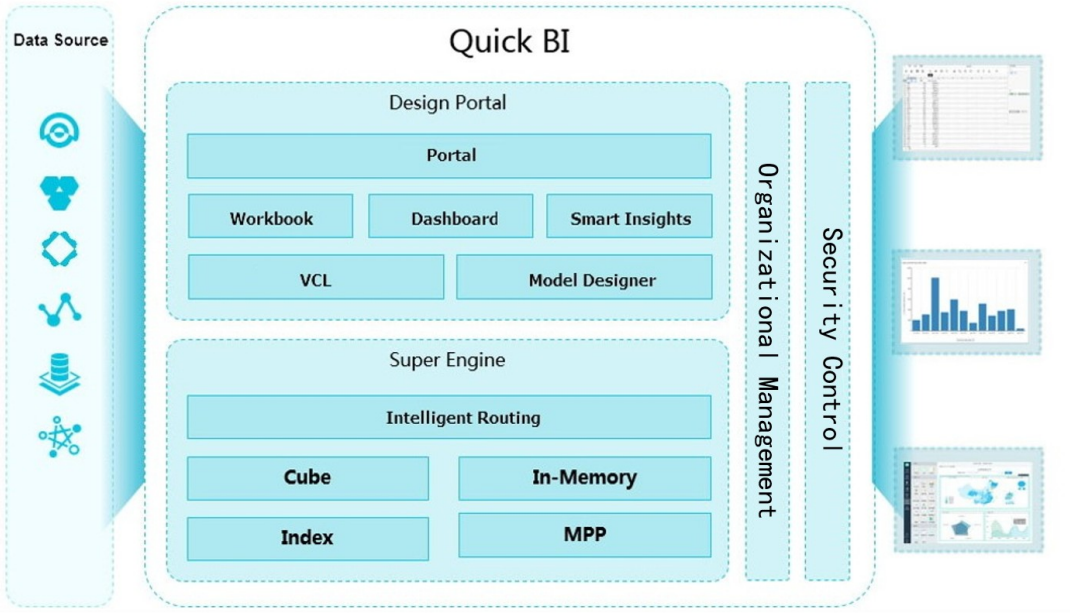
Quick BI is a flexible and lightweight self-service Business Intelligence (BI) tool based on cloud computing. It supports a wide range of data sources:

- MaxCompute (formerly known as ODPS) and ApsaraDB for RDS
- User-created MySQL and SQL Server databases that are hosted on ECS
- VPC data sources

Quick BI analyzes large amounts of data in real time and returns results within seconds. You do not need to preprocess the data. Quick BI analyzes terabytes of incremental data on a daily basis.

With an intelligent data modeling tool and a variety of visual chart tools, Quick BI significantly reduces data acquisition costs and makes it easier for you to use Quick BI features. This allows you to easily complete data analysis, self-service data acquisition, business data query, and report making.

Architecture



7.2. Log on to the Quick BI console

This topic describes how to log on to the Quick BI console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].

- We recommend that you use the Google Chrome browser.

Context

If you are using a RAM account, you can use the domain name to log on to the Quick BI console. If you are using an Apsara Stack tenant account, log on to the Quick BI console by performing the following steps:

Procedure

1. In the address bar, enter the access address of the Apsara Stack Cloud Management (ASCM) console, and press Enter.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the homepage of the ASCM console.
4. In the top navigation bar, choose **Product** > **Big Data** and click **Quick BI**.
5. Select an organization and a region from the drop-down lists and click **QuickBI** or **Quick BI Console** to go to the **Quick BI** system.
   - You can click **QuickBI** to go to the product page.
   - You can click **Quick BI Console** to go to the settings page. You can manage organization members and workspaces on this page.

     > **Note** If you use this method, the Apsara Stack tenant account quickbi_admin@aliyun.com is used to log on to the Quick BI console no matter which department you select.

7.3. Data modeling

7.3.1. Data modeling

Data modeling visualizes data and allows you to quickly identify and extract information. It also helps you make correct decisions based on data trends.

Steps of data modeling:

1. Create a data source.
2. Create a dataset.

7.3.2. Data sources

7.3.2.1. Overview of data sources

Datasets, workbooks, dashboards, and BI portals are all created based on data sources. Quick BI supports both cloud data sources and user-created data sources.

Cloud data sources include:

- MaxCompute
- MySQL
- SQL Server
- AnalyticDB for MySQL 2.0
- HybridDB for MySQL
- AnalyticDB for PostgreSQL
- PostgreSQL
- PPAS
- Hive
- Data Lake Analytics
- Distribute Relational Database Service (DRDS)
- Presto
- AnalyticDB for MySQL 3.0
- PolarDB for MySQL

> Note VPCs only support MySQL and SQL Server data sources.

User-created data sources include:

- MySQL
- SQL Server
- PostgreSQL
- Oracle
- Hive
- Vertica
- IBM DB2 LUW
- SAP IQ (SybaseIQ)
- SAP HANA
- Presto

> Note You cannot check SQL Server data sources by using views.

7.3.2.2. Cloud data sources

7.3.2.2.1. Add the IP addresses of a Quick BI cluster to a database whitelist

This topic describes how to add the IP addresses of a Quick BI cluster to a database whitelist.

Prerequisites

The Quick BI service is purchased.

Context

Before you add some data sources in the Quick BI console, you must first query available IP addresses of machines and add them to a whitelist of ApsaraDB for RDS.

Procedure

1. Log on to the Apsara Infrastructure Management Framework console.Make sure that you have obtained the URL of the Apsara Stack Operations (ASO) console, and the username and password used for logging on to the console from a deployment engineer or administrator.
   i. Open a browser, enter the URL in the address bar, and press Enter. The URL is in the format of *region-id*.aso.*intranet-domain-id*.com.

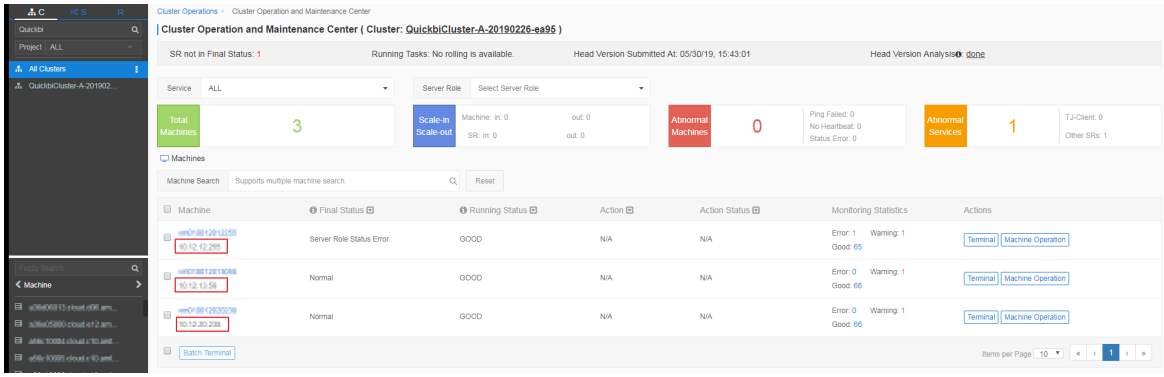
      > **Note** We recommend that you use the Google Chrome browser.
      >
      > You can select a language from the drop-down list in the upper-right corner.

   ii. Enter the correct username and password.
      - The following user roles are available by default:
        - security administrator: the user who has the permissions to manage other users or roles
        - auditor: the user who has the permissions to view audit logs
        - sysadmin: the user who has the permissions that security administrators and auditors do not have
      - When you log on to the ASO console for the first time, you must change the password of your username as prompted.

        For security concerns, your password must contain the following characters:
        - Letters
        - Digits
        - Special characters, such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

        The password must be 10 to 20 characters in length.

   iii. Click **Log On** to go to the **ASO** console.
   iv. In the left-side navigation pane, choose **Products** > **Product List**. Click **Apsara Infrastructure Management Framework**.
2. On the homepage of the Apsara Infrastructure Management Framework console, enter the keywords of the name of the target Quick BI cluster in the search box, select the target cluster from the drop-down list, and then click **Operations** next to the cluster.
3. You can view the IP addresses of the Quick BI cluster on the **Machines** tab.



What's next

Add the IP addresses to an ApsaraDB for RDS whitelist.

Before you add the IP addresses to the ApsaraDB for RDS whitelist, change the last octet of all the addresses to 0/24. For example, if an IP address is 10.10.10.10, change it to 10.10.10.0/24. For information about how to add an IP address to an ApsaraDB for RDS whitelist, see "Configure a whitelist" in ApsaraDB for RDS User Guide.

7.3.2.2.2. Add a cloud MaxCompute data source

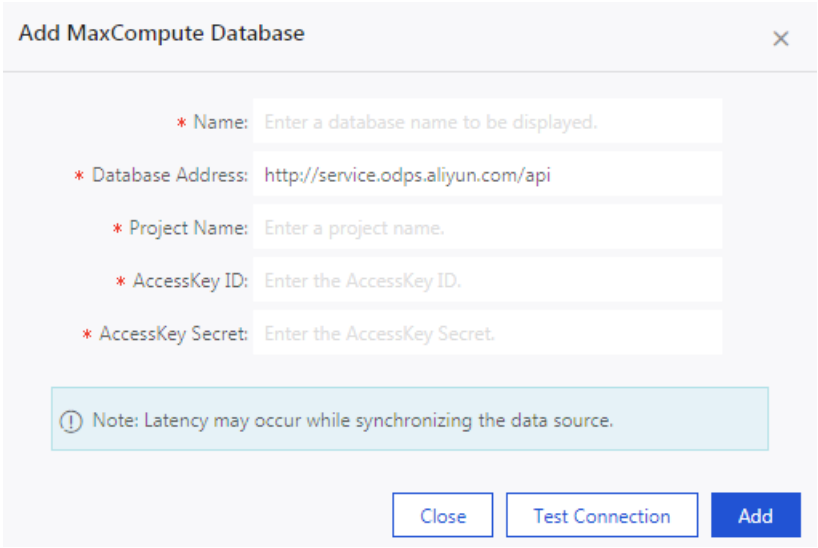
This topic describes how to add a cloud MaxCompute data source.

Prerequisites

- A MaxCompute (formerly known as ODPS) project is created in the big data computing service console.
- Parameter settings for connecting to the MaxCompute data source are obtained.

Procedure

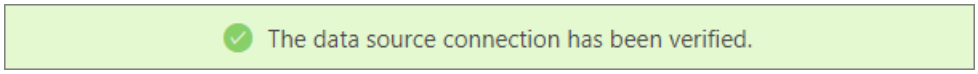
1. Create a data source.
2. Click **MaxCompute**. In the **Add MaxCompute Database** dialog box that appears, set the following parameters.



3. Set the parameters for connecting to the data source.

Parameter	Description
Name	The name of the data source that you want to display.
Database Address	The default database address.
Project Name	The name of the MaxCompute project.
AccessKey ID	The AccessKey ID used to identify a visitor.
AccessKey Secret	The AccessKey secret of the account that purchased the data source instance. The AccessKey secret is used to encrypt the signature string on the client and decrypt the signature string on the server for authentication. Keep the AccessKey secret confidential.

4. Click **Test Connection** to perform a data source connectivity test.



5. After the connection is established, click **Add**.

   After the data source is added, the Data Sources page appears. Tables under the data source are listed on the right side of the page.

7.3.2.2.3. Add a cloud data source ApsaraDB RDS for MySQL

This topic describes how to add a cloud data source ApsaraDB RDS for MySQL.

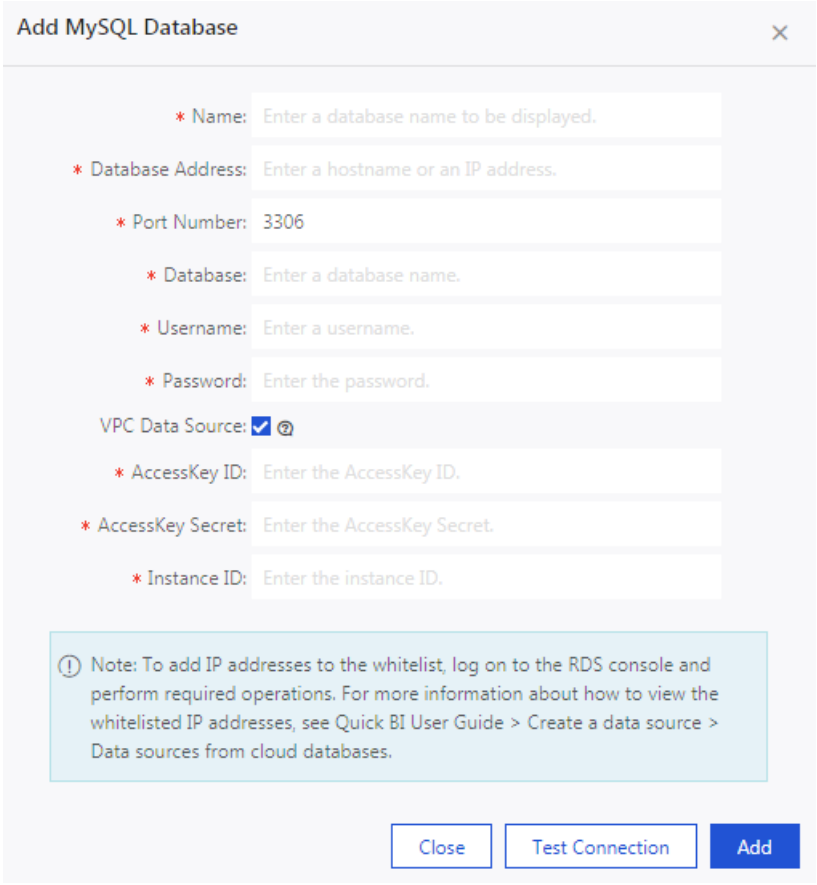
Prerequisites

1. An ApsaraDB RDS for MySQL instance is created. For information about how to create an ApsaraDB for RDS instance, see "Create an instance" in ApsaraDB for RDS User Guide.
2. Available IP addresses of machines are obtained and added to a whitelist of the ApsaraDB for RDS instance. For information about how to query the IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the ApsaraDB for RDS instance, see "Configure a whitelist" in ApsaraDB for RDS User Guide.
3. Parameter settings used to connect to the cloud data source ApsaraDB RDS for MySQL are obtained.

Procedure

1. Create a data source.
2. Click the **MySQL** card and configure the required parameters in the dialog box that appears.

   Add a cloud data source ApsaraDB RDS for MySQL



> **Note** If the data source is connected over a VPC, select **VPC Data Source** and configure the required parameters.

Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database. Default value: 3306.
Database	The name of the database that you want to access.
Username	The username that you use to access the database.
Password	The password that you use to access the database. > **Note** If you do not know the username and password, contact the data warehouse administrator.
AccessKey ID	The AccessKey ID that you use to purchase the database instance.

Parameter	Description
AccessKey Secret	The AccessKey secret that you use to purchase the database instance.
Instance ID	The ID of the database instance.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

   After a data source is added, you cannot add it again. If you attempt to add a data source for the second time, an error message appears.

7.3.2.2.4. Add a cloud data source ApsaraDB RDS for SQL Server

This topic describes how to add a cloud data source ApsaraDB RDS for SQL Server.

Prerequisites

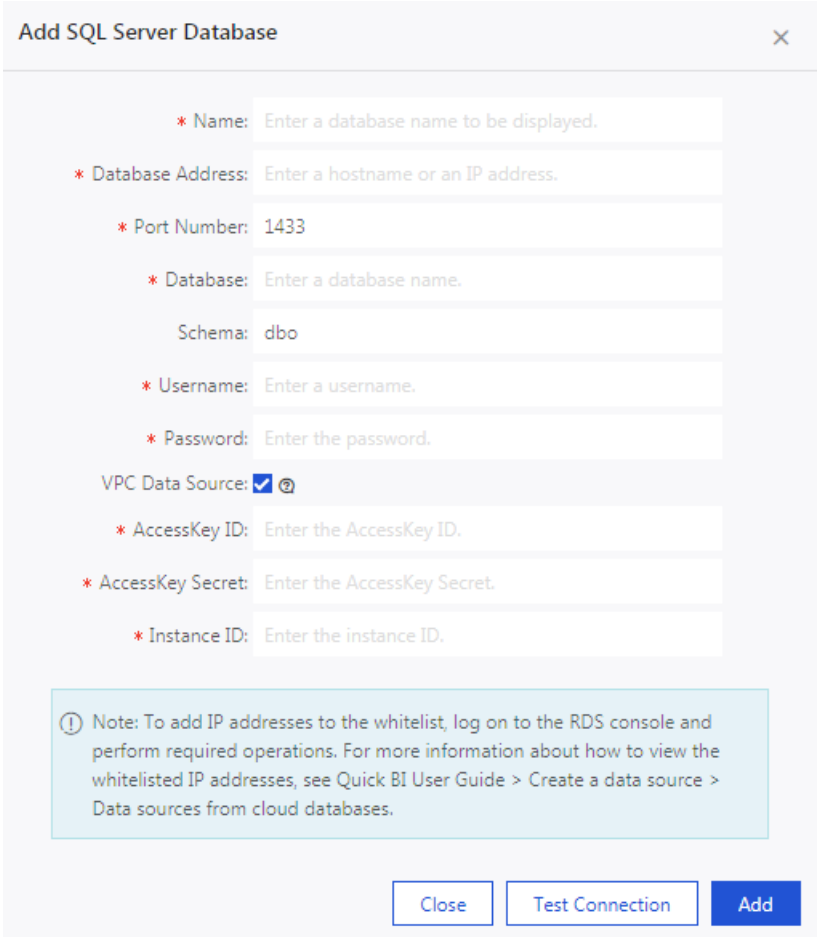
1. An ApsaraDB RDS for SQL Server instance is created. For information about how to create an ApsaraDB for RDS instance, see "Create an instance" in ApsaraDB for RDS User Guide.
2. Available IP addresses of machines are obtained and added to a whitelist of the ApsaraDB for RDS instance. For information about how to query the IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the ApsaraDB for RDS instance, see "Configure a whitelist" in ApsaraDB for RDS User Guide.
3. Parameter settings used to connect to the cloud data source ApsaraDB RDS for SQL Server are obtained.

Context

The method of using an ApsaraDB RDS for SQL Server database as a data source is similar to that of using an ApsaraDB RDS for MySQL database as a data source. The only difference is that you must set the Schema parameter when you use an ApsaraDB RDS for SQL Server database as a data source.

Procedure

1. Create a data source.
2. Click the **SQL Server** card and configure the required parameters in the dialog box that appears.



Note If the data source is connected over a VPC, select **VPC Data Source** and configure the required parameters.

Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database. Default value: 1433.
Database	The name of the database that you want to access.
Schema	The database schema. Default value: dbo.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
AccessKey ID	The AccessKey ID that you use to purchase the database instance.

Parameter	Description
AccessKey Secret	The AccessKey secret that you use to purchase the database instance.
Instance ID	The ID of the database instance.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.2.5. Add an AnalyticDB for MySQL 2.0 data source

This topic describes how to add an AnalyticDB for MySQL 2.0 data source.

Prerequisites

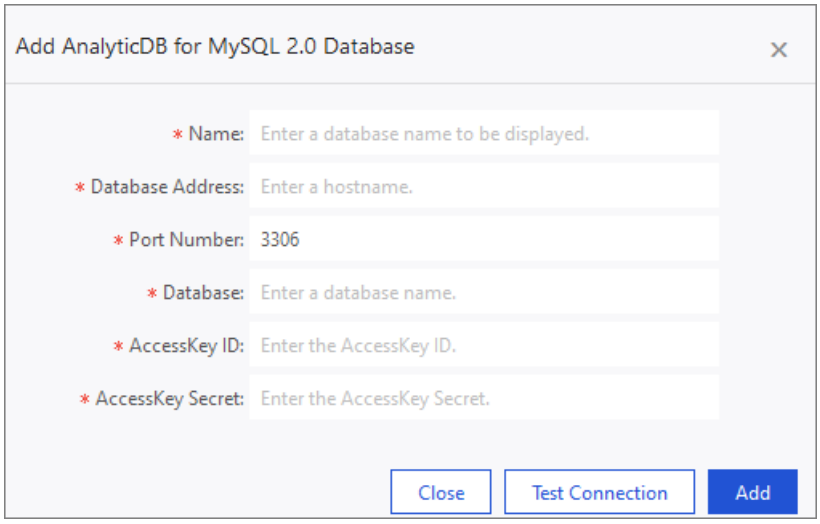
An AnalyticDB database is created in the AnalyticDB console, and parameter settings used to connect to the AnalyticDB data source are obtained. For information about how to create an AnalyticDB database, see "Create a database" in AnalyticDB for MySQL User Guide.

Context

AnalyticDB for MySQL is formerly known as AnalyticDB.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the Cloud Data Sources tab. Then, click **AnalyticDB for MySQL 2.0** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.

Parameter	Description
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database.
Database	The name of the database to which you want to access.
AccessKey ID	The AccessKey ID used to identify a visitor.
AccessKey Secret	The AccessKey secret is used to encrypt the signature string on the client and decrypt the signature string on the server for authentication. Keep the AccessKey secret confidential.

3. Click **Test Connection** to test connectivity with the data source.
4. After the connection is established, click **Add**.

7.3.2.2.6. Add a cloud HybridDB for MySQL data source

This topic describes how to add a cloud HybridDB for MySQL data source.

Prerequisites

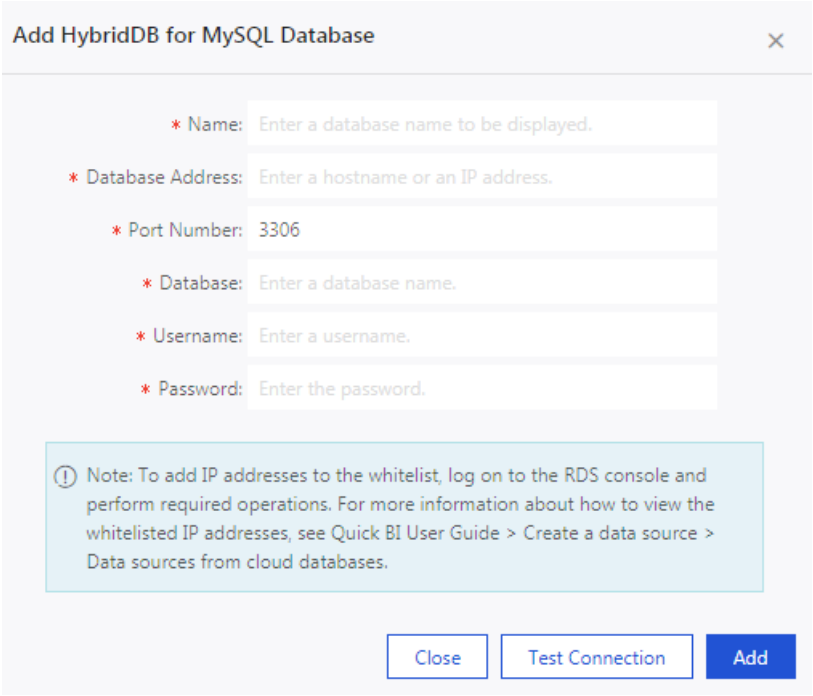
1. A HybridDB for MySQL instance is created.
2. The available machine IP addresses are obtained and added to a whitelist of the HybridDB for MySQL instance. For information about how to query the available IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist.
3. Parameter settings used to connect to the HybridDB for MySQL data source are obtained.

Context

The procedure for adding a HybridDB for MySQL data source is similar to that for adding an SQL Server data source. The only difference is that the default port is the port specific to HybridDB for MySQL.

Procedure

1. Create a data source.
2. Click **HybridDB for MySQL** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 3306.
Database	The name of the database to which you want to access.
Username	The username used to access the database.
Password	The password of the database user.

3. Click **Test Connection** to test the data source connectivity.
4. After the connection is established, click **Add**.

7.3.2.2.7. Add a cloud AnalyticDB for PostgreSQL data source

This topic describes how to add a cloud AnalyticDB for PostgreSQL data source.

Prerequisites

1. An AnalyticDB for PostgreSQL instance is created. For information about how to create an AnalyticDB for PostgreSQL instance, see "Create an instance" in AnalyticDB for PostgreSQL User Guide.
2. The available machine IP addresses are obtained and added to a whitelist of the AnalyticDB

for PostgreSQL instance. For information about how to query the available IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the AnalyticDB for PostgreSQL instance, see "Configure a whitelist" in AnalyticDB for PostgreSQL User Guide.

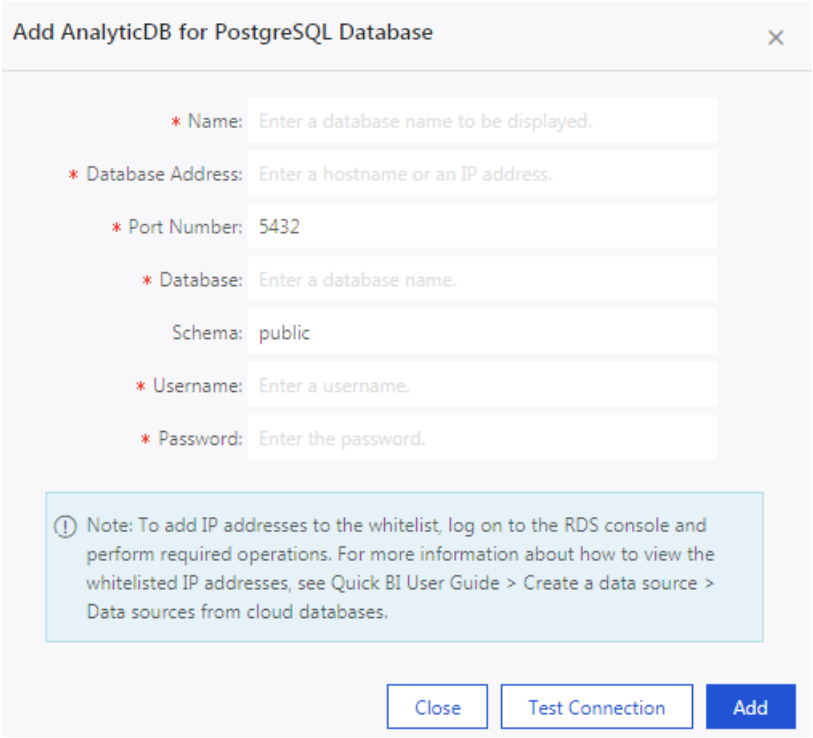
3. Parameter settings used to connect to the AnalyticDB for PostgreSQL data source are obtained.

Context

The procedure for adding an AnalyticDB for PostgreSQL data source is similar to that for adding an SQL Server data source. The only difference is that the default port is the port specific to AnalyticDB for PostgreSQL.

Procedure

1. Create a data source.
2. Click **AnalyticDB for PostgreSQL** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 5432.
Database	The name of the database to which you want to access.
Schema	The default value is Public.

Parameter	Description
Username	The username used to access the database.
Password	The password of the database user.

3. Click **Test Connection** to test the data source connectivity.
4. After the connection is established, click **Add**.

7.3.2.2.8. Add a cloud data source AnalyticDB for PostgreSQL

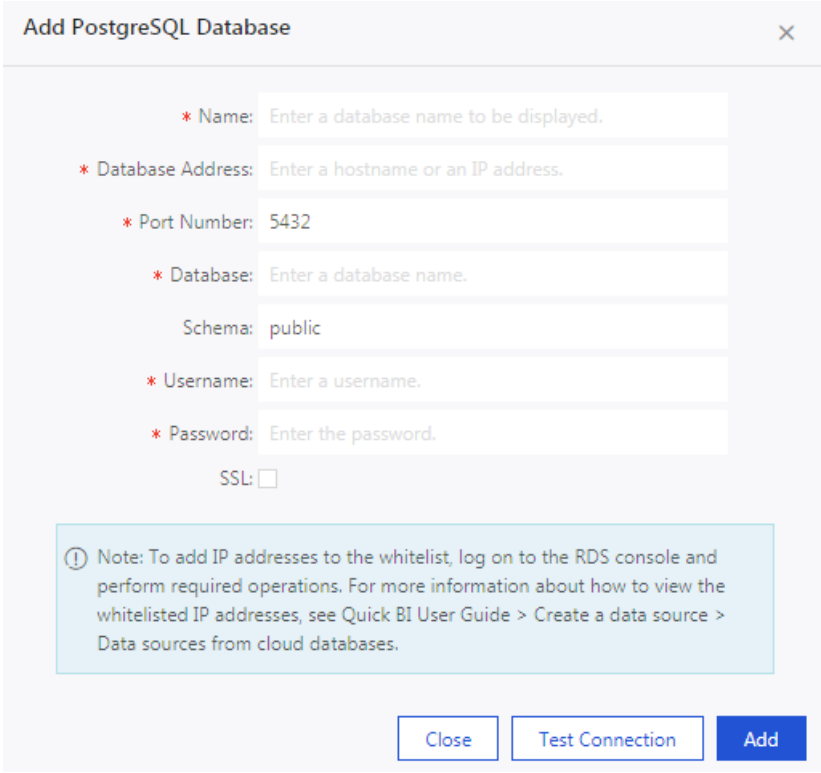
This topic describes how to add a cloud data source AnalyticDB for PostgreSQL.

Prerequisites

1. An RDS for PostgreSQL instance is created. For information about how to create an ApsaraDB for RDS instance, see "Create an instance" in ApsaraDB for RDS User Guide.
2. Available IP addresses of machines are obtained and added to a whitelist of the ApsaraDB for RDS instance. For information about how to query the IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the ApsaraDB for RDS instance, see "Configure a whitelist" in ApsaraDB for RDS User Guide.
3. Parameter settings used to connect to a cloud data source AnalyticDB for PostgreSQL are obtained.

Procedure

1. Create a data source.
2. Click the **PostgreSQL** card and configure the required parameters in the dialog box that appears.



Note After you select the **SSL** option, the data source supports the interactive query service MaxCompute Lightning provided by MaxCompute.

Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database. Default value: 5432.
Database	The name of the database that you want to access.
Schema	The database schema. Default value: public.
Username	The username that you use to access the database.
Password	The password that you use to access the database.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.2.9. Add a cloud PPAS data source

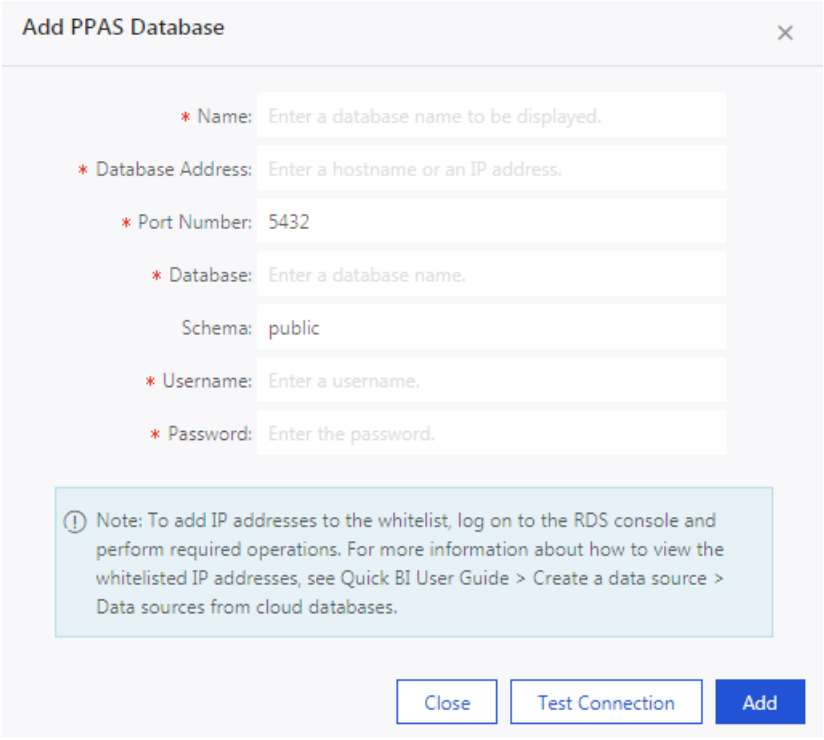
This topic describes how to add a cloud PPAS data source.

Prerequisites

1. An RDS for PPAS instance is created. For information about how to create an ApsaraDB for RDS instance, see "Create an instance" in ApsaraDB for RDS User Guide.
2. The available IP addresses are obtained and added to a whitelist of the ApsaraDB for RDS instance. For information about how to query the IP addresses, see Add the IP addresses of a Quick BI cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the ApsaraDB for RDS instance, see "Configure a whitelist" in ApsaraDB for RDS User Guide.
3. Parameter settings used to connect to the PPAS data source are obtained.

Procedure

1. Create a data source.
2. Click **PPAS** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port	The port number of the database. The default port is 5432.
Database	The name of the database to which you want to access.
Schema	The default value is Public.

Parameter	Description
Username	The username used to access the database.
Password	The password of the database user.

3. Click **Test Connection** to test the data source connectivity.
4. After the connection is established, click **Add**.

7.3.2.2.10. Add a cloud Hive data source

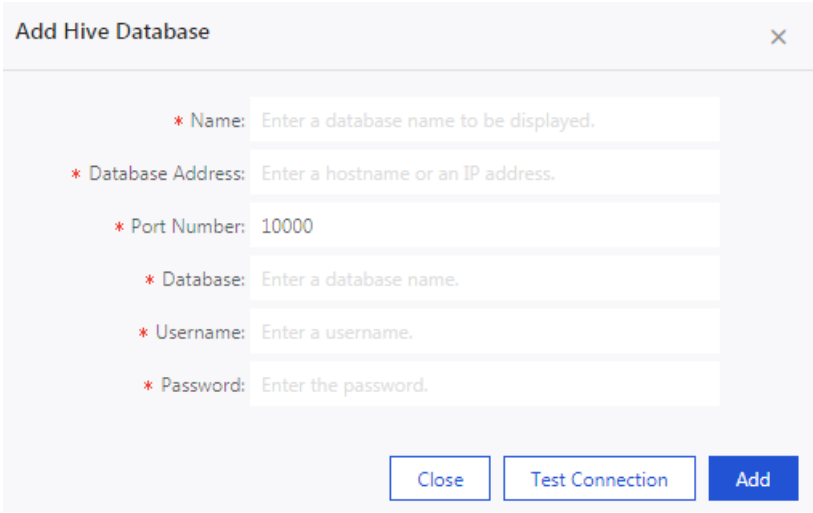
This topic describes how to add a cloud Hive data source.

Prerequisites

Parameter settings used to connect to the Hive data source are obtained.

Procedure

1. Create a data source.
2. Click **Hive** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 10000.
Database	The name of the database to which you want to access.
Username	The username used to access the database.

Parameter	Description
Password	The password of the database user.

3. Click **Test Connection** to test the data source connectivity.
4. After the connection is established, click **Add**.

7.3.2.2.11. Add a cloud data source Data Lake Analytics

This topic describes how to add a cloud data source Data Lake Analytics (DLA).

Prerequisites

Parameter settings used to connect to the cloud data source DLA are obtained.

Procedure

1. Create a data source.
2. Click the **Data Lake Analytics** card and configure the required parameters in the dialog box that appears.

Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database. Default value: 10000.
Database	The name of the database that you want to access.
Username	The username that you use to access the database.
Password	The password that you use to access the database.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.2.12. Add a cloud DRDS data source

This topic describes how to add a cloud Distributed Relational Database Service (DRDS) data source.

Prerequisites

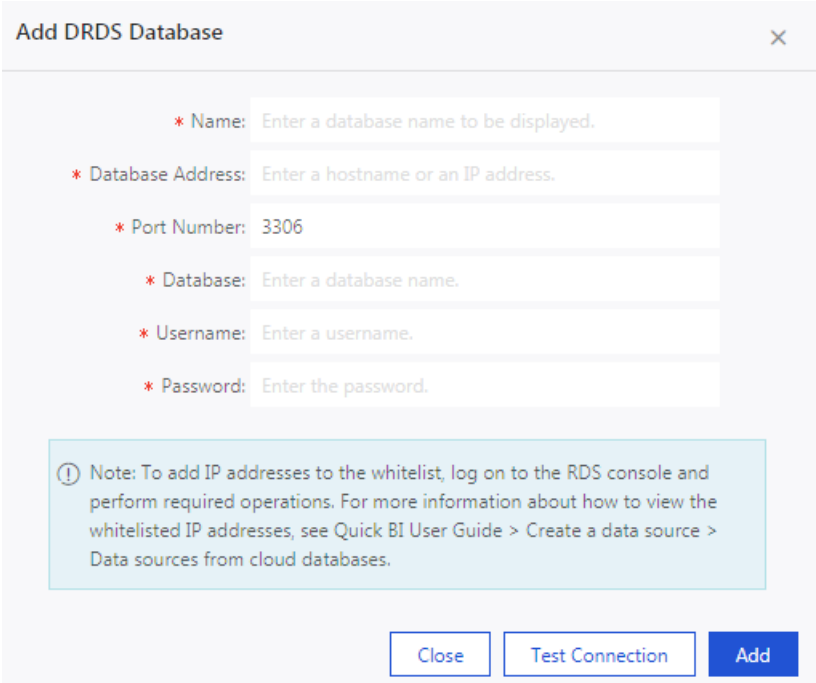
1. A DRDS instance is created. For information about how to create a DRDS instance, see "Create an instance" in ApsaraDB for RDS User Guide.
2. The available IP addresses are obtained and added to a whitelist of the DRDS instance. For information about how to query the IP addresses, see Add the IP addresses of a Quick BI

cluster to a database whitelist. For information about how to add the IP addresses to a whitelist of the DRDS instance, see "Configure a whitelist" in ApsaraDB for RDS User Guide.

3. Parameter settings used to connect to the DRDS data source are obtained.

Procedure

1. Create a data source.
2. Click **DRDS** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 3306.
Database	The name of the database to which you want to access.
Username	The username used to access the database.
Password	The password of the database user.

3. Click **Test Connection** to test the data source connectivity.
4. After the connection is established, click **Add**.

7.3.2.2.13. Add a cloud Presto data source

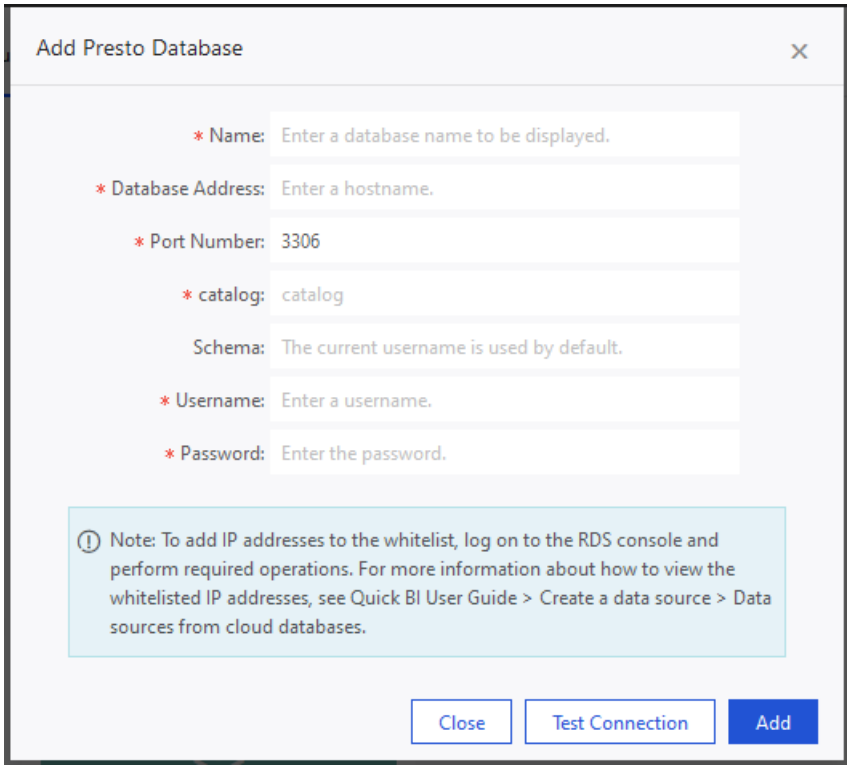
This topic describes how to add a cloud Presto data source.

Prerequisites

Parameter settings used to connect to the Presto data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the Cloud Data Sources tab. Then, click **Presto** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 3306.
catalog	The name of the data source that you want to use Presto to query.
Schema	The current logon username is used by default.
Username	The username used to access the database.
Password	The password used to access the database.

3. Click **Test Connection** to test connectivity with the database.

4. After the connection is established, click **Add**.

7.3.2.2.14. Add a cloud data source AnalyticDB for MySQL V3.0

This topic describes how to add a cloud data source AnalyticDB for MySQL V3.0.

Prerequisites

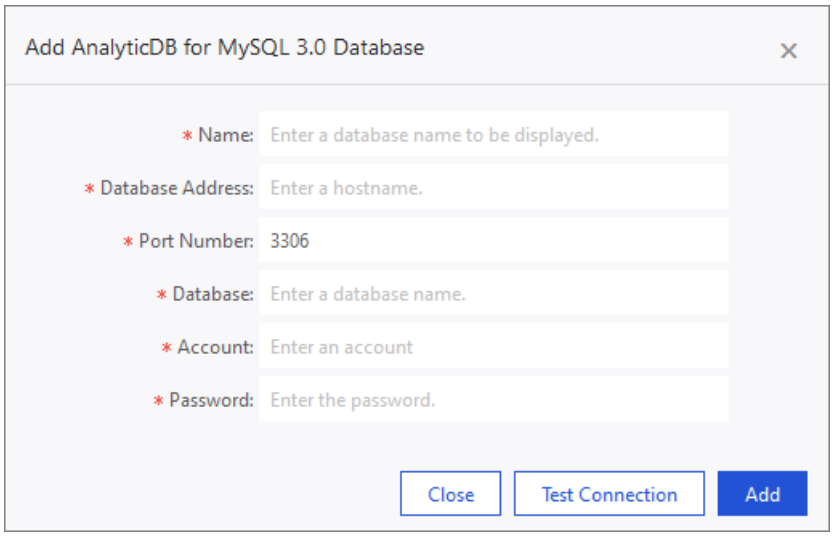
An AnalyticDB for MySQL V3.0 database is created in the AnalyticDB for MySQL console, and parameter settings for connecting to the database are obtained. For information about how to create an AnalyticDB for MySQL V3.0 database, see "Create a database" in AnalyticDB for MySQL User Guide.

Context

AnalyticDB for MySQL is formerly known as AnalyticDB.

Procedure

1. Create a data source.
2. Click the **AnalyticDB for MySQL 3.0** card and configure the required parameters in the dialog box that appears.



Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database.
Database	The name of the database that you want to access.
Username	The username that you use to access the database.

Parameter	Description
Password	The password that you use to access the database.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.2.15. Add a cloud data source PolarDB for MySQL

This topic describes how to add a cloud data source PolarDB for MySQL.

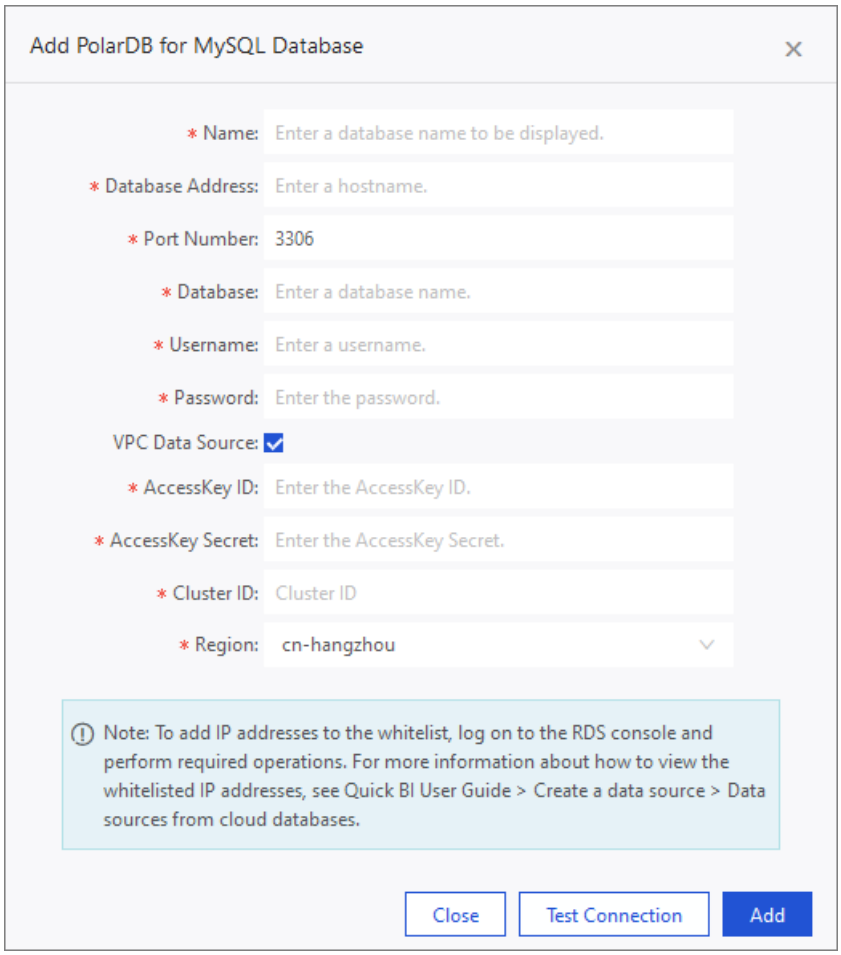
Prerequisites

Parameter settings used to connect to the cloud data source PolarDB for MySQL are obtained.

Procedure

1. Create a data source.
2. Click the **PolarDB for MySQL** card and configure the required parameters in the dialog box that appears.

> **Note** If the data source is connected over a VPC, select VPC Data Source and configure the required parameters.



Parameter	Description
Name	The name of the data source, which is user-defined.
Database Address	The hostname or IP address of the database that is used as the data source.
Port Number	The port that you use to access the database. Default value: 3306.
Database	The name of the database that you want to access.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
AccessKey ID	The AccessKey ID that you use to purchase the database instance.
AccessKey Secret	The AccessKey secret that you use to purchase the database instance.
Cluster ID	The ID of the cluster.
Region	The region where the instance resides.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.3. User-created data sources

7.3.2.3.1. Add a user-created MySQL data source

This topic describes how to add a user-created MySQL data source.

Prerequisites

Parameter settings used to connect to the MySQL database are obtained.

Context

The procedure for configuring a user-created MySQL data source is similar to that for configuring an ApsaraDB RDS for MySQL data source. You must perform the following operations to open the specific port of the firewall to allow Quick BI to access the MySQL database:

1. Run the following command to open the configuration file of the firewall:

```
vi /etc/sysconfig/iptables
```

2. Add the following command to the configuration file:

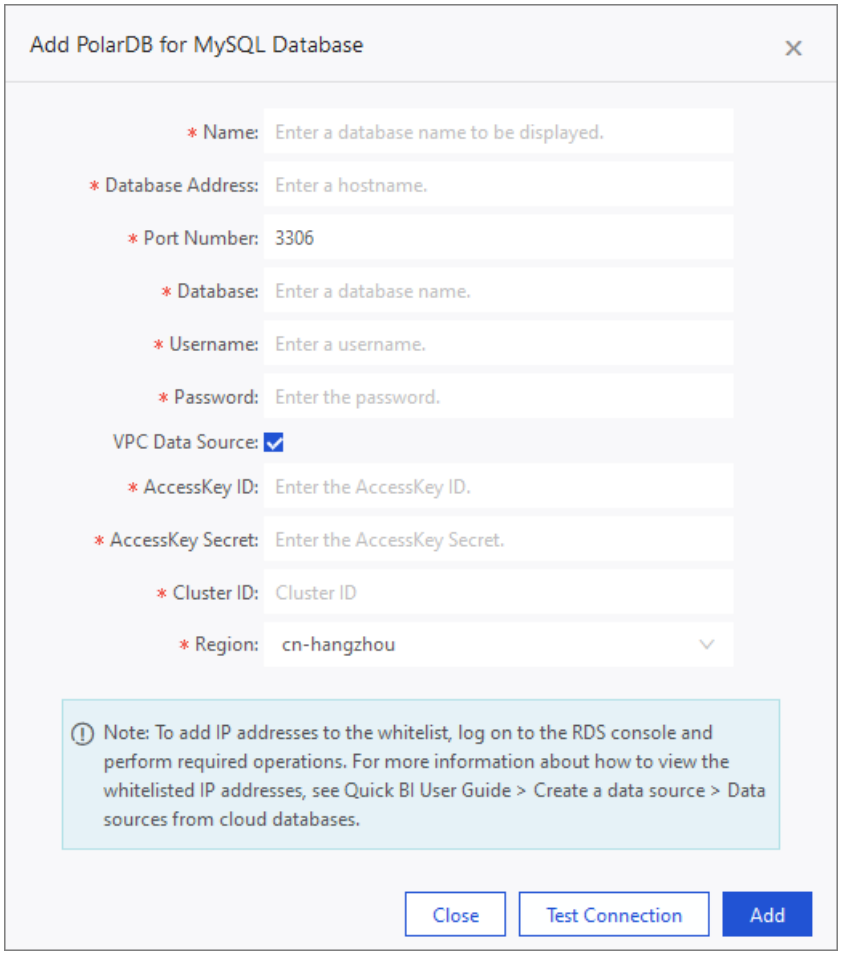
```
-A RH-Firewall-1-INPUT -m state -state NEW -m tcp -p tcp -dport 3306 -j ACCEPT
```

3. Restart the iptables service after the configuration is complete.

```
service iptables restart
```

Procedure

1. Go to the Add Data Source page. For more information, see Add a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **MySQL** and configure the parameters for connecting to the database in the dialog box that appears.



> **Note** If you select ssh, you can access the database by using an SSH tunnel.

Parameter	Description
Name	The name of the data source.
Database Address	The hostname or IP address of the database.
Port	The port that you use to access the database. Default value: 3306.
Database	The name of the database that you want to access.
Username	The username that you use to access the database.

Parameter	Description
Password	The password that you use to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username that you use to access the SSH host.
SSH Password	The password that you use to access the SSH host.
SSH Port Number	The port number that you use to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.3.2. Add a user-created SQL Server data source

This topic describes how to add a user-created SQL Server data source.

Prerequisites

Parameter settings used to connect to the SQL Server database are obtained.

Context

The procedure for configuring a user-created SQL Server data source is similar to that for configuring an ApsaraDB RDS for SQL Server data source. You must perform the following operations to open the specified port of the firewall to allow Quick BI to access the SQL Server database:

1. Run the following command to open the configuration file of the firewall:

   ```
   vi /etc/sysconfig/iptables
   ```

2. Add the following command to the configuration file:

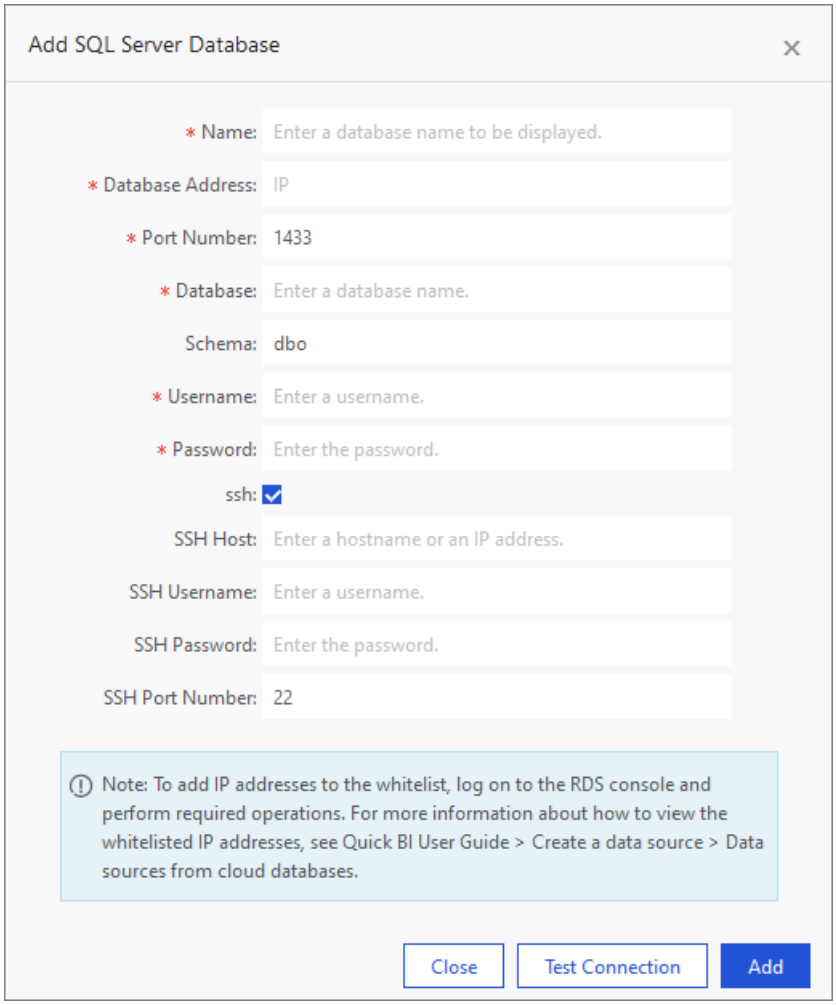
   ```
   -A RH-Firewall-1-INPUT -m state -state NEW -m tcp -p tcp -dport 3306 -j ACCEPT
   ```

3. After the configuration is complete, run the following command to restart the iptable service:

   ```
   service iptables restart
   ```

Procedure

1. Go to the Add Data Source page. For more information, see Add a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click the **SQL Server** card and configure the parameters for connecting to the database in the dialog box that appears.



Note If you select ssh, you can access the data source by using an SSH tunnel.

Parameter	Description
Name	The name of the data source.
Database Address	The hostname or IP address of the database.
Port	The port that you use to access the database. Default value: 1433.
Database	The name of the database that you want to access.
Schema	The database schema. Default value: dbo.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username that you use to access the SSH host.

Parameter	Description
SSH Password	The password that you use to access the SSH host.
SSH Port Number	The port number that you use to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

> **Note** You cannot add a data source that has been added. If you attempt to add it again, an error message appears.

7.3.2.3.3. Add a user-created PostgreSQL data source

This topic describes how to add a user-created PostgreSQL data source.

Prerequisites

Parameter settings used to connect to the PostgreSQL database are obtained.

Context

The procedure for configuring a user-created PostgreSQL data source is similar to that for configuring an ApsaraDB RDS for PostgreSQL data source. You must perform the following operations to open the specified port of the firewall to allow Quick BI to access the PostgreSQL database:

1. Run the following command to open the configuration file of the firewall:

```
vi /etc/sysconfig/iptables
```

2. Add the following command to the configuration file:

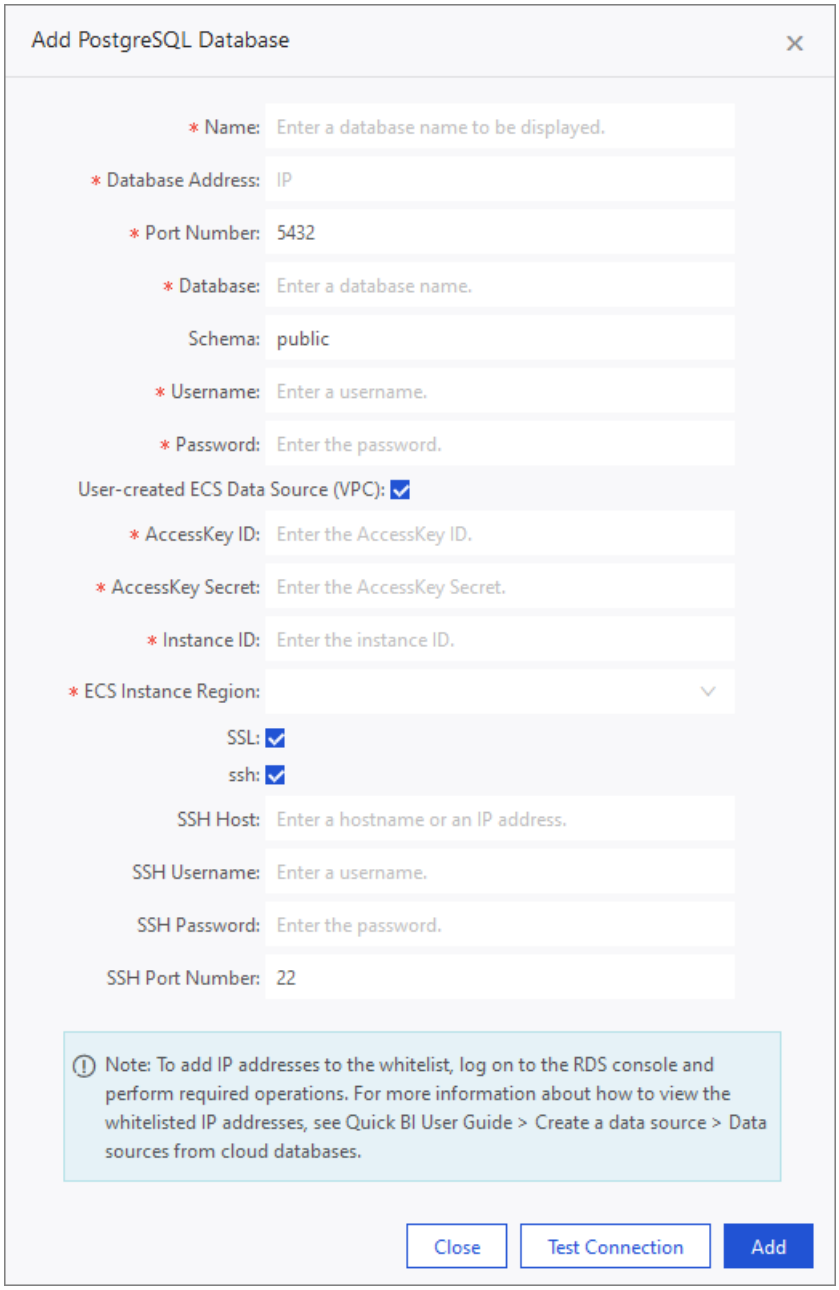
```
-A RH-Firewall-1-INPUT -m state -state NEW -m tcp -p tcp -dport 3306 -j ACCEPT
```

3. After the configuration is complete, run the following command to restart the iptable service:

```
service iptables restart
```

Procedure

1. Go to the Add Data Source page. For more information, see Add a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click the **PostgreSQL** card and configure the parameters for connecting to the database in the dialog box that appears.



Note If the data source is connected over a VPC, select **User-created ECS Data Source (VPC)** and specify required parameters. If you select **SSL**, MaxCompute Lightning is supported. If you select **ssh**, you can access the data source by using an SSH tunnel.

Parameter	Description
Name	The name of the data source.
Database Address	The hostname or IP address of the database.
Port	The port that you use to access the database. Default value: 5432.

Parameter	Description
Database	The name of the database that you want to access.
Schema	The database schema. Default value: public.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
AccessKey ID	The AccessKey ID that you use to purchase the database instance.
AccessKey Secret	The AccessKey secret that you use to purchase the database instance.
Instance ID	The ID of the database instance.
ECS Instance Region	The region where the data source is located.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username that you use to access the SSH host.
SSH Password	The password that you use to access the SSH host.
SSH Port Number	The port number that you use to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.3.4. Add a user-created Oracle data source

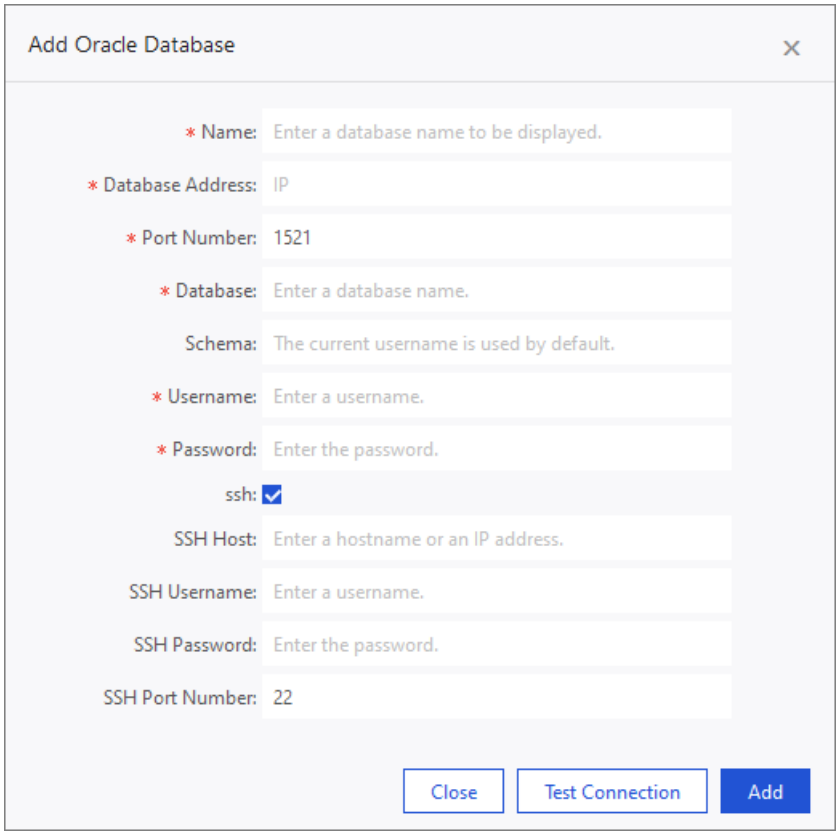
This topic describes how to add a user-created Oracle data source.

Prerequisites

Parameter settings used to connect to the Oracle database are obtained.

Procedure

1. Go to the Add Data Source page. For more information, see Add a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click the **Oracle** card and configure the parameters for connecting to the database in the dialog box that appears.



Note If you select ssh, you can access the data source by using an SSH tunnel.

Parameter	Description
Name	The name of the data source.
Database Address	The hostname or IP address of the database.
Port	The port that you use to access the database. Default value: 1521.
Database	The name of the database that you want to access.
Schema	The database schema. Default value: public.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username that you use to access the SSH host.
SSH Password	The password that you use to access the SSH host.
SSH Port Number	The port number that you use to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.3.5. Add a user-created Hive data source

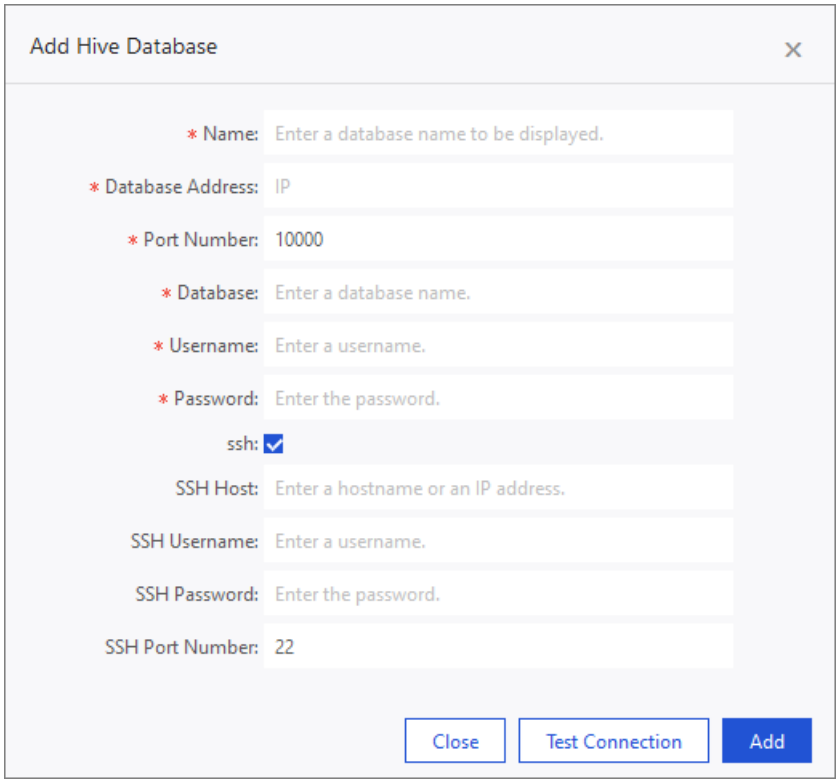
This topic describes how to add a user-created Hive data source.

Prerequisites

Parameter settings for connecting to the Hive data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **Hive** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 10000.
Database	The name of the database to which you want to access.

Parameter	Description
Username	The username used to access the database.
Password	The password used to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username used to access the SSH host.
SSH Password	The password used to access the SSH host.
SSH Port Number	The port number used to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the connection is established, click **Add**.

7.3.2.3.6. Add a user-created Vertica data source

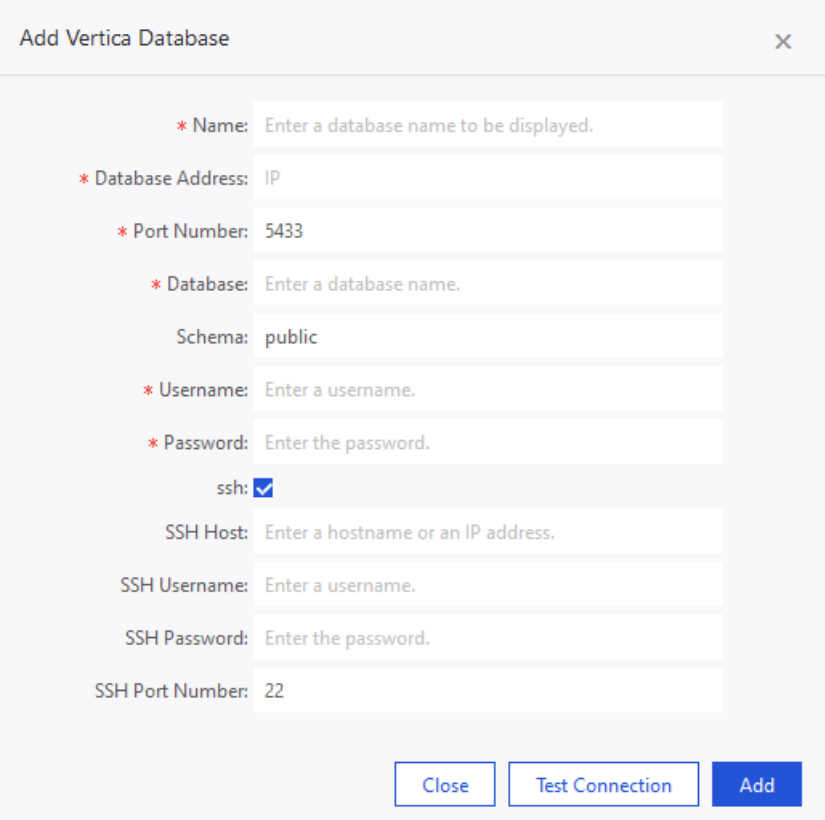
This topic describes how to add a user-created Vertica data source.

Prerequisites

Parameter settings for connecting to the Vertica data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **Vertica** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 5433.
Database	The name of the database to which you want to access.
Schema	The schema of the database. The default value is public.
Username	The username used to access the database.
Password	The password used to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username used to access the SSH host.
SSH Password	The password used to access the SSH host.
SSH Port Number	The port number used to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the connection is established, click **Add**.

7.3.2.3.7. Add a user-created IBM DB2 LUW data source

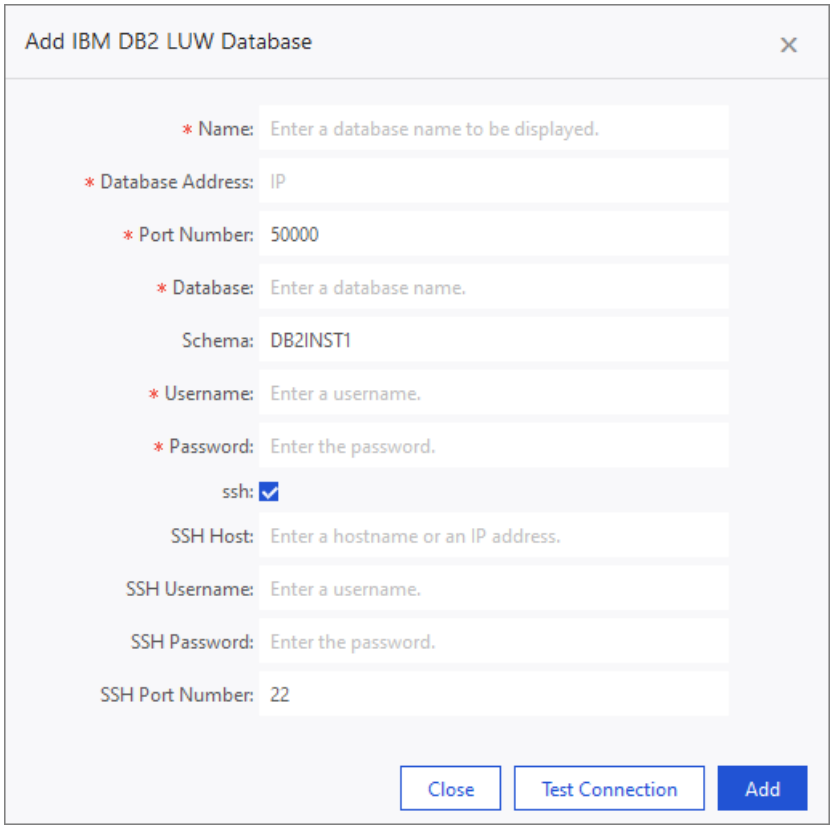
This topic describes how to add a user-created IBM DB2 LUW data source.

Prerequisites

Parameter settings for connecting to the IBM DB2 LUW data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **IBM DB2 LUW** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 50000.
Database	The name of the database that you want to access.
Schema	The schema of the database. The default value is DB2INST1.

Parameter	Description
Username	The username used to access the database.
Password	The password used to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username used to access the SSH host.
SSH Password	The password used to access the SSH host.
SSH Port Number	The port number used to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the data source.
4. After the connection is established, click **Add**.

7.3.2.3.8. Add a user-created SAP IQ (Sybase IQ) data source

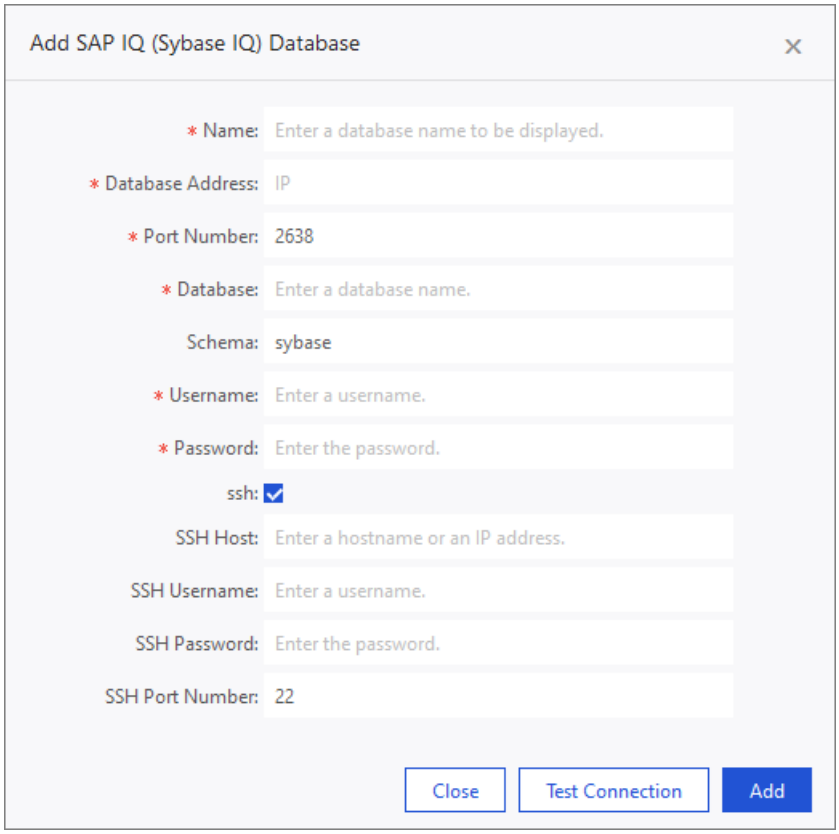
This topic describes how to add a user-created SAP IQ (Sybase IQ) data source.

Prerequisites

Parameter settings for connecting to the SAP IQ (Sybase IQ) data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **SAP IQ (Sybase IQ)** and set the parameters for connecting to the data source in the dialog box that appears.



Parameter	Description
Name	The name to be displayed for the connected data source.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 2638.
Database	The name of the database that you want to access.
Schema	The schema of the database. The default value is sybase.
Username	The username used to access the database.
Password	The password used to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username used to access the SSH host.
SSH Password	The password used to access the SSH host.
SSH Port Number	The port number used to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the data source.
4. After the connection is established, click **Add**.

7.3.2.3.9. Add a user-created SAP HANA data source

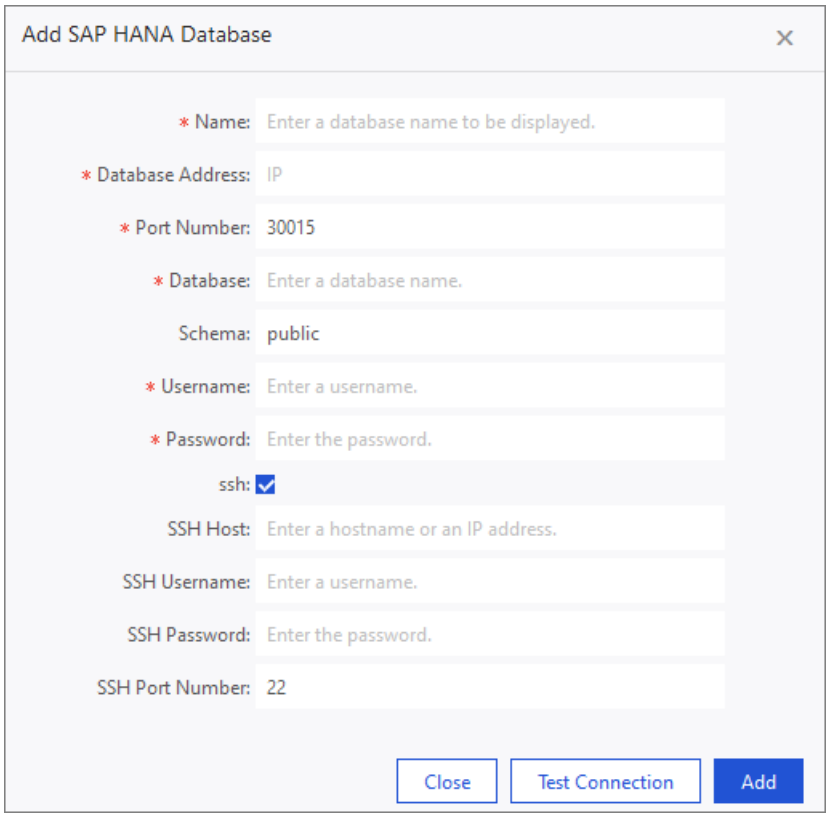
This topic describes how to add a user-created SAP HANA data source.

Prerequisites

Parameter settings used to connect to the SAP HANA database are obtained.

Procedure

1. Go to the Add Data Source page. For more information, see Add a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click the **SAP HANA** card and configure the parameters for connecting to the database in the dialog box that appears.



> **Note** If you select ssh, you can access the database by using an SSH tunnel.

Parameter	Description
Name	The name of the data source.
Database Address	The hostname or IP address of the database.
Port	The port that you use to access the database. Default value: 30015.
Database	The name of the database that you want to access.

Parameter	Description
Schema	The database schema. Default value: public.
Username	The username that you use to access the database.
Password	The password that you use to access the database.
SSH Host	The hostname or IP address of the SSH host.
SSH Username	The username that you use to access the SSH host.
SSH Password	The password that you use to access the SSH host.
SSH Port Number	The port number that you use to access the SSH host. Set the value to 22.

3. Click **Test Connection** to test connectivity with the database.
4. After the database passes the connectivity test, click **Add**.

7.3.2.3.10. Add a user-created Presto data source

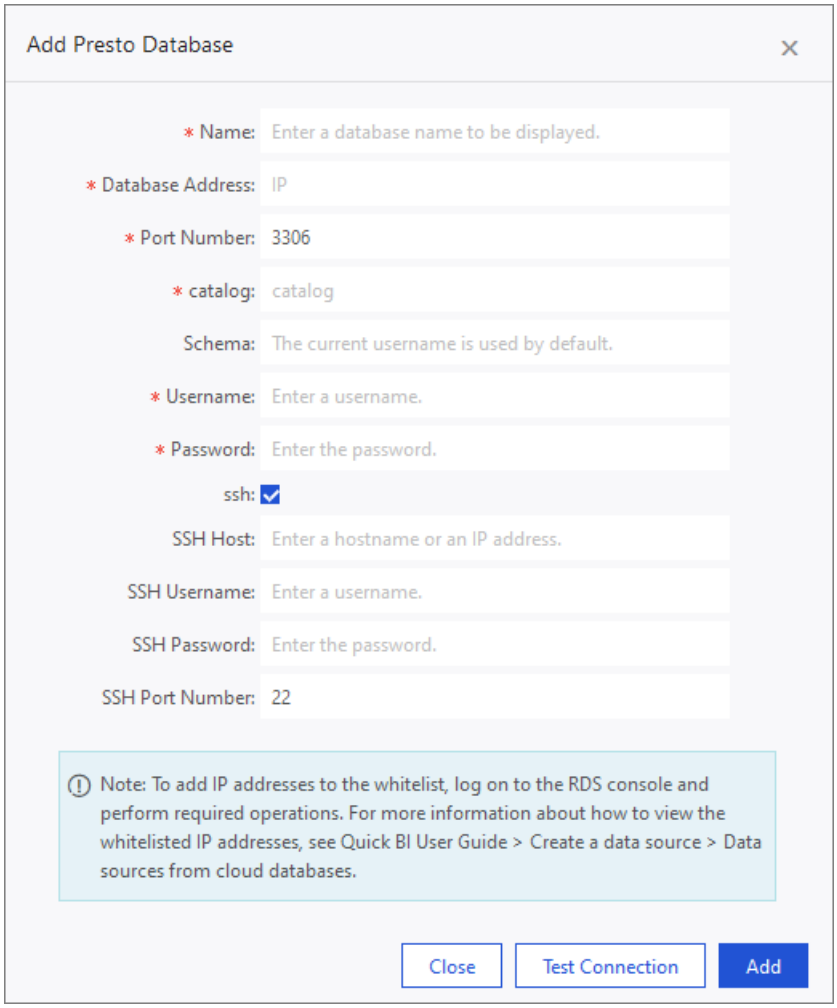
This topic describes how to add a user-created Presto data source.

Prerequisites

Parameter settings for connecting to the Presto data source are obtained.

Procedure

1. On the Data Sources page, click Create Data Source in the upper-right corner. For more information, see Create a data source.
2. On the Add Data Source page, click the User-created Data Sources tab. Then, click **Presto** and set the parameters for connecting to the data source in the dialog box that appears.



Note If you select ssh, you can access the data source by using an SSH tunnel.

Parameter	Description
Name	The name to be displayed for the connected data source, which is user-defined.
Database Address	The hostname or IP address of the database.
Port Number	The port number of the database. The default port is 3306.
catalog	The name of the data source that you want to use Presto to query.
Schema	The current logon username is used by default.
Username	The username used to access the database.
Password	The password used to access the database.
SSH Host	The hostname or IP address of the SSH host.

Parameter	Description
SSH Username	The username used to access the SSH host.
SSH Password	The password used to access the SSH host.
SSH Port Number	The port number used to access the SSH host. Set the value to 22.

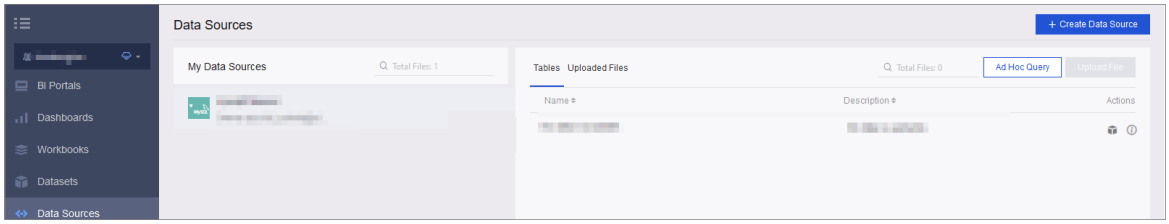
3. Click **Test Connection** to test connectivity with the data source.
4. After the connection is established, click **Add**.

7.3.2.4. List of data sources

This topic describes the basic information about the Data Sources page.

1. Log on to the Quick BI console.
2. Click the Workspace tab. In the left-side navigation pane, click **Data Sources**.

   On this page, you can manage data sources. For example, you can **create**, **edit**, and **delete** a data source.



7.3.2.5. Create a data source

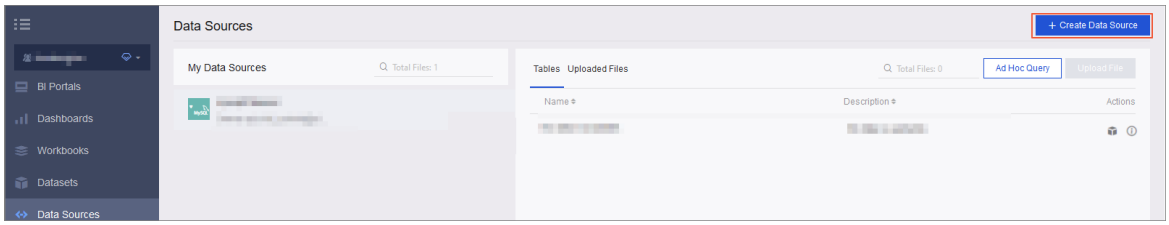
This topic describes how to create a data source.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Data Sources**.
4. In the upper-right corner of the Data Sources page, click **Create Data Source**.



5. In the **Add Data Source** dialog box that appears, select a data source.
   - If you want to add a cloud data source, click the **Cloud Data Sources** tab.

- If you want to add a user-created data source, click the **User-created Data Sources** tab.

6. Select the data source type. In the dialog box that appears, enter the information required for connecting to the data source, and click **Add**.

7.3.2.6. Edit a data source

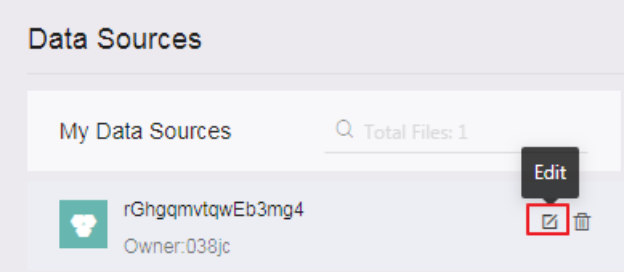
This topic describes how to edit a data source.

Prerequisites

- The Quick BI service is purchased.
- A data source is created.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.
4. Find the data source that you want to edit in the data source list.
5. Click the icon and edit the data source.



7.3.2.7. Delete a data source

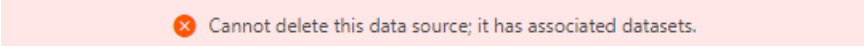
This topic describes how to delete a data source.

Prerequisites

- The Quick BI service is purchased.
- A data source is created.

Context

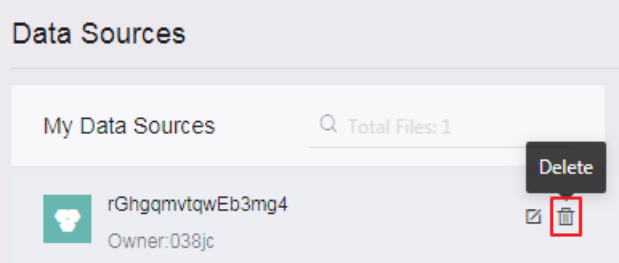
If you have created datasets based on a data source, you cannot delete the data source.



Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.

4. Find the data source that you want to delete in the data source list.
5. Click the icon to delete the data source.



7.3.2.8. Search for a data source

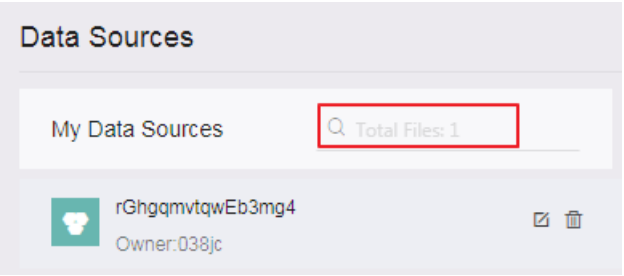
This topic describes how to search for a data source.

Prerequisites

- The Quick BI service is purchased.
- A data source is created.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.
4. On the Data Sources page, enter a keyword of the data source that you want to search for in the search box.



5. Click the icon.

7.3.2.9. Search for a table under a data source

This topic describes how to search for a specific table under a data source.

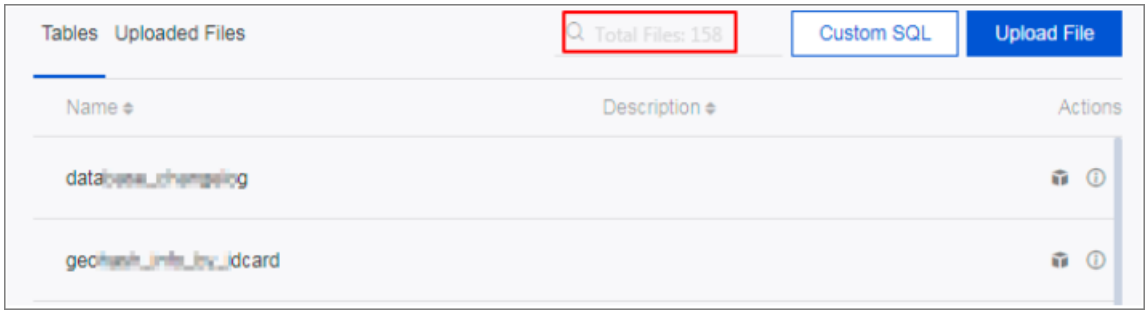
Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.

3. In the left-side navigation pane, click **Data Sources**.
4. Select the target data source. All tables under the data source are listed in the right-side part of the page.
5. Enter a keyword of the table that you want to search in the search box.



6. Click the icon.

7.3.2.10. View the details of a table under a data source

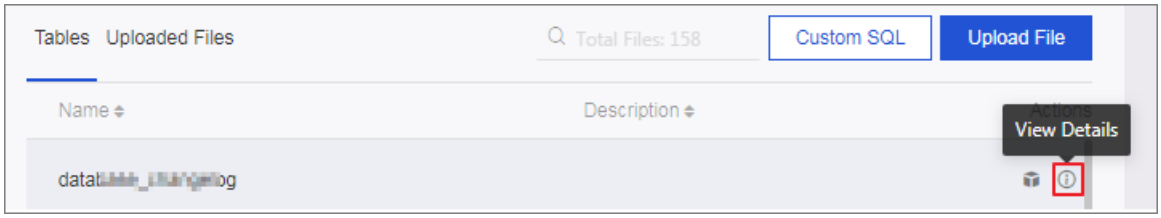
Quick BI allows you to view the tables and table details under a data source.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.
4. Select the target data source. All tables under the data source are listed in the right-side part of the page.
5. Find the table whose details you want to view and click the icon in the Actions column.



7.3.3. Datasets

7.3.3.1. Overview of datasets

You can use tables from data sources to create datasets. The following topics describe common operations on datasets, for example, create, edit, and query a dataset.

You can use one of the following three methods to create a dataset:

- Create a dataset from a data source.

- Create a dataset by uploading a file.
- Create a dataset by using an SQL statement for ad hoc query.

7.3.3.2. Create datasets

7.3.3.2.1. Create a dataset based on a data source

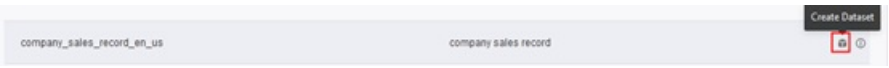
This topic describes how to create a dataset based on a data source.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.
4. On the Data Sources page, click the target data source. All tables under the data source are listed in the right-side part of the page.
5. Find the target table and click the icon in the Actions column.



After the dataset is created, the **Datasets** page appears. The new dataset is displayed on the My Items tab and marked with NEW.

7.3.3.2.2. Create a dataset by uploading a file

This topic describes how to create a dataset by uploading a file.

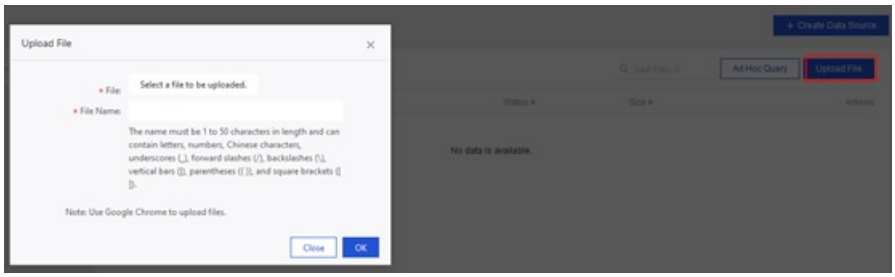
Prerequisites

- The Quick BI service is purchased.
- A data source is created.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Data Sources**.

4. On the Data Sources page, click a data source, and then click **Upload File** in the upper-right corner of the page.
5. In the **Upload File** dialog box, select the target file, enter a file name, and then click **OK**.



> **Note** After the file is uploaded, the Uploaded Files page appears.

6. On the Uploaded Files page, find the file you uploaded and click the icon in the Actions column to create a dataset.

7.3.3.2.3. Create a dataset by using an SQL statement for ad hoc query

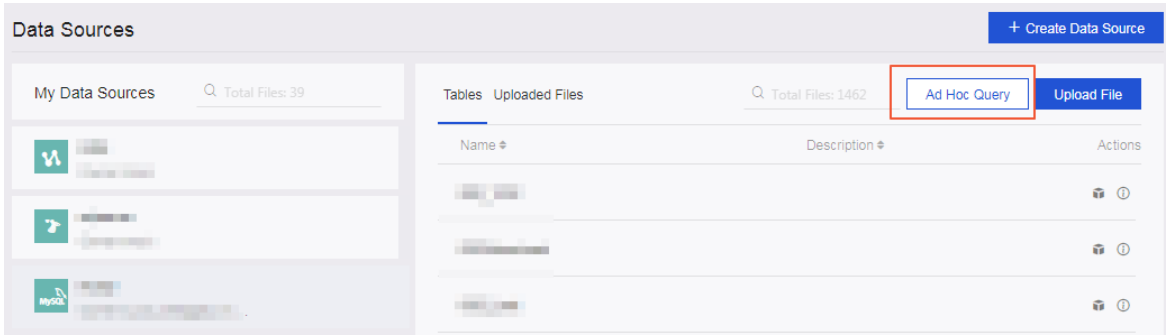
This topic describes how to create a dataset by using an SQL statement for ad hoc query to implement some complex logic for data modeling. Ad hoc queries support dynamic parameter passing to SQL statements. Modeling analysis based on dynamic parameter passing to SQL statements increases the depth of scenarios supported by Quick BI. This meets the requirements of complex data analysis scenarios.

Prerequisites

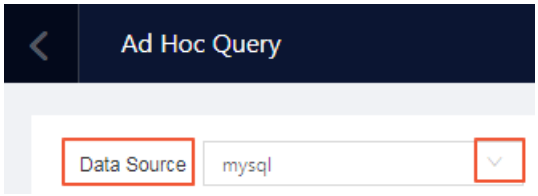
The Quick BI service is purchased.

Create a dataset by using an SQL statement for ad hoc query

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Data Sources**.
4. In the upper-right corner of the **Data Sources** page, click **Ad Hoc Query**.



5. On the **Ad Hoc Query** page that appears, specify the data source.



6. Enter an SQL statement.

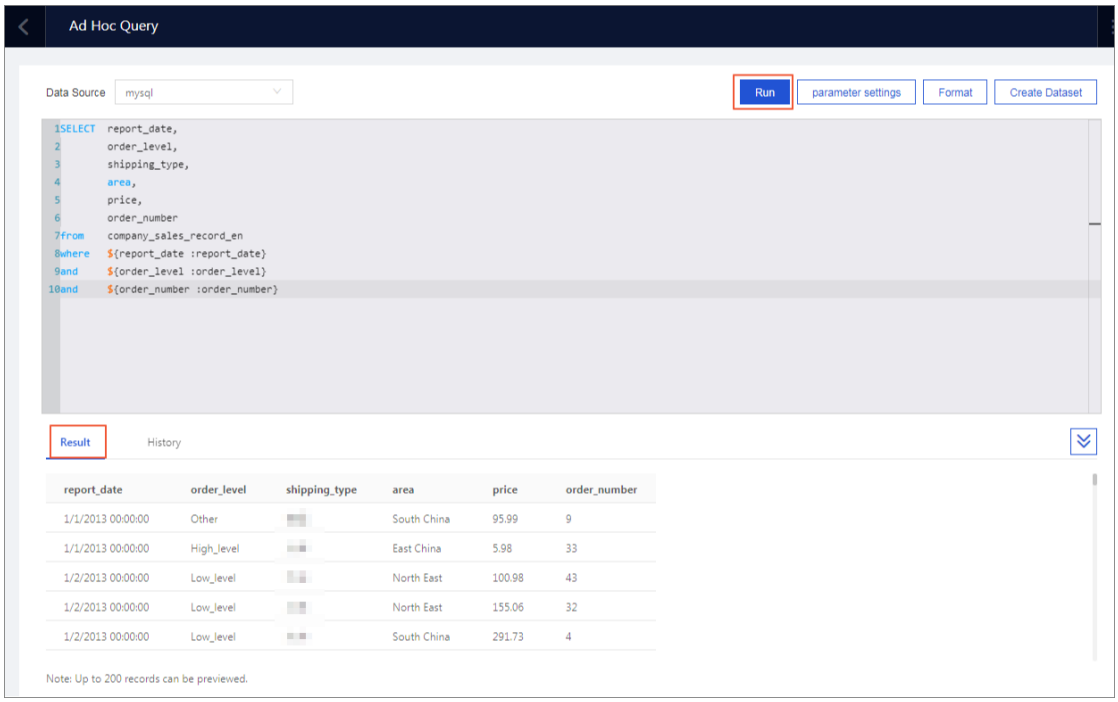
   Sample SQL statement:

```
SELECT  report_date,
        order_level,
        shipping_type,
        area,
        price,
        order_number
from    company_sales_record
where   ${report_date :report_date}
and     ${order_level :order_level}
and     ${order_number :order_number}
```

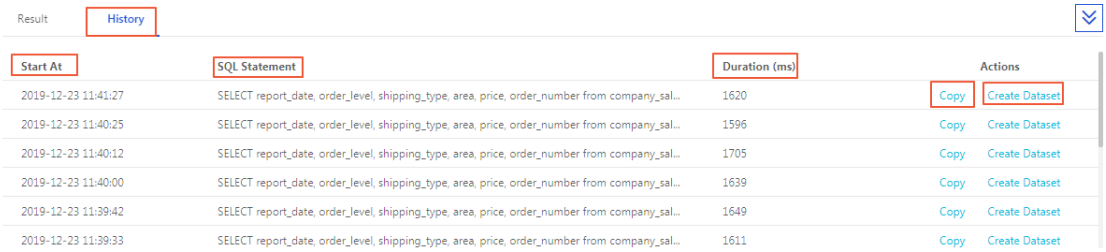
7. Click **Run** to execute the SQL statement.

   You can view the execution results on the Result tab.

   i. Click the **Result** tab.



ii. Click the **History** tab to view the execution time, SQL statement, and time consumed by the ad hoc query.



- Click **Copy** to copy the SQL statement and paste it into the SQL statement input box.
- Click **Create Dataset**. A dataset is created by using the historical SQL statement.
- Click to hide the execution results.

SQL statements for ad hoc queries support dynamic parameter passing and placeholders. When you create an SQL model, you can append SQL parameters to the WHERE clause in the format of $ {Physical field name:Parameter alias}. The parameters can be referenced by the filter bar widget.

> **Note** Parameter fields are not displayed in the dataset, but are displayed in the filter bar widget.

Sample SQL statement:

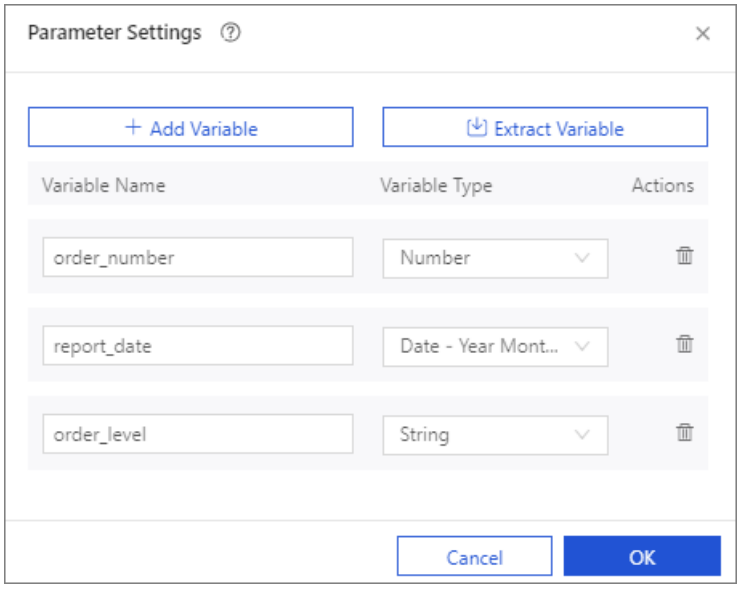
```
SELECT  report_date,
      order_level,
      shipping_type,
      area,
      price,
      order_number
from   company_sales_record
where   ${report_date :report_date}
and     ${order_level :order_level}
and     ${order_number :order_number}
```

8. Before you execute SQL statements, you can set and format parameters. After the SQL statements are executed, you can create datasets.

i. Click **parameter settings**. In the dialog box that appears, configure the parameters.

   You can add variables and modify variable types. Five variable types are supported: String, Number, Date -Year Month Date, Date -Year Month, and Date -Year.

   - Click **New parametric variables** to add parameter aliases and variable types. A parameter alias must be added to the WHERE clause in the SQL statement in the format of ${Physical field name:Parameter variable name}.
   - Click **Fast extraction** to obtain the parameter aliases in the SQL statement. The default variable type is String, which can be changed.



ii. Click **Format** to format the SQL statement.

iii. Click **Create Dataset** to create a dataset based on the query results of the current SQL statement.

For information about how to use the parameters in the filter bar widget, see Use SQL statement parameters in the Filter Bar widget.

7.3.3.3. Specify a method to name dimensions and measures

Quick BI creates datasets based on the metadata of physical tables and converts fields in the physical tables to dimensions or measures in the datasets. The names of dimensions and measures are automatically generated. They can be the names or descriptions of the fields in the physical tables.

Prerequisites

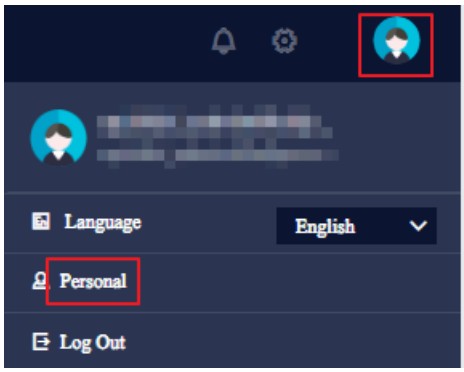
The Quick BI service is purchased.

Context

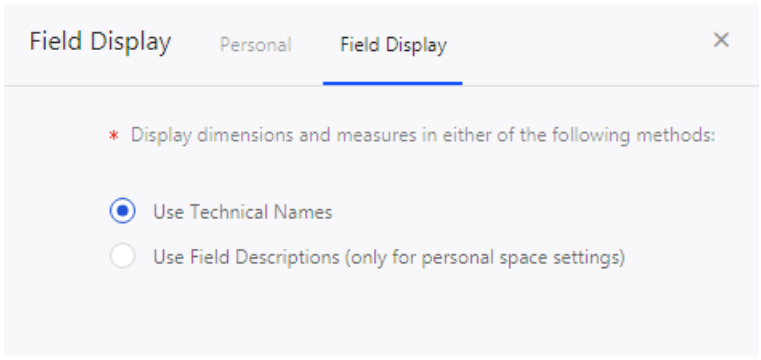
You can move the pointer over the profile picture and specify a method to name dimensions and measures in datasets. Dimensions and measures in datasets are named based on your setting.

Procedure

1. Log on to the Quick BI console.
2. Move the pointer over the profile picture in the upper-right corner and select **Personal**.



3. In the **Personal** dialog box, click the **Display** tab and select a method to name dimensions and measures in datasets.



7.3.3.4. Edit a dataset

7.3.3.4.1. Edit a dimension

If the data type of a field in a dataset table is String or Date/Time, the field is classified as a dimension by the system. You can edit the field in the Dimensions list.

Prerequisites

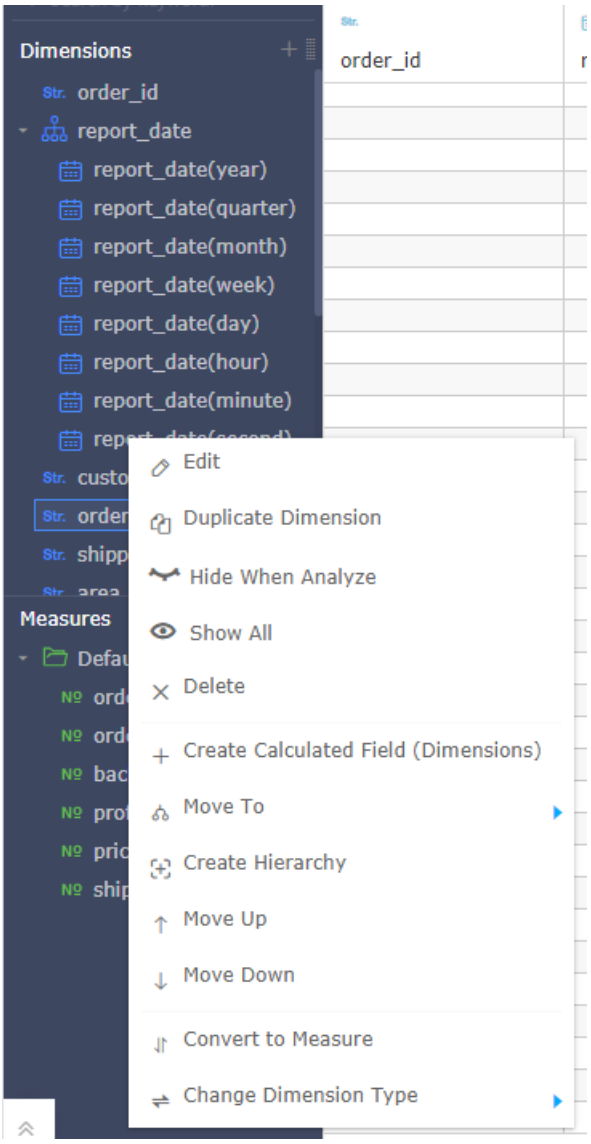
The Quick BI service is purchased.

A dataset is created.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the Datasets page, find the target dataset and click the icon in the Actions column to go to the dataset edit page.
5. Find the target field in the Dimensions list.

6. Right-click the field and select the operation you want to perform.



- **Edit**: Edit the name and description of the dimension.

  When you edit a week dimension, you can configure the first day of a week.
- **Duplicate Dimension**: Duplicate the dimension. The name of the duplicate dimension ends with **Duplicate**.
- **Hide When Analyze**: Hide the dimension as required.
- **Show All**: Show all dimensions.
- **Delete**: Delete the dimension.
- **Create Calculated Field (Dimensions)**: Create a dimension and customize its calculation method.
- **Move To**: Move the dimension to an existing hierarchy for drilling.
- **Create Hierarchy**: Add the dimension field to a new hierarchy.
- **Move Up/Move Down**: Move the dimension. You can drag or right-click the dimension to move it.

- **Convert to Measure**: Convert the dimension to a measure.
- **Change Dimension Type**: Switch the dimension type among Date/Time (Source Format), Geo, String, and Number.

  For example, you must change the data type of the province and city dimensions to Geo when you create a bubble map or colored map. Otherwise, the map cannot be created.

  > **Note** You can duplicate, hide, and delete dimensions at different hierarchy levels.

7.3.3.4.2. Edit a measure

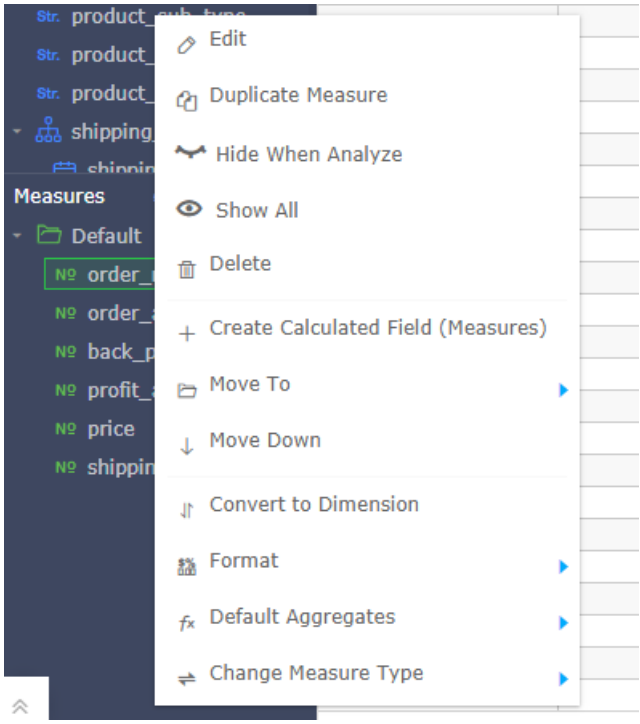
If the data type of a field in a dataset table is Number, the field is classified as a measure by default. You can edit the field in the Measures list.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the Datasets page, click the target dataset to go to the dataset edit page.
5. Find the target field in the Measures list.
6. Right-click the field and select the operation that you want to perform on the field as required.



- Edit: Edit the name and description of the measure.
- Duplicate Measure: Duplicate the measure. The name of the duplicate measure ends with Duplicate.
- Hide When Analyze: Hide the measure as required.
- Show All: Show all measures.
- Delete: Delete the measure.
- Create Calculated Field (Measures): Create a measure and customize its calculation method.
- Move To: Move the measure to an existing folder.
- Move Up/Move Down: Move the measure. You can drag or right-click the measure to move it.
- Convert to Dimension: Convert the measure to a dimension.
- Format: Specify the number format.
- Default Aggregates: Specify the aggregate function, such as SUM, MAX, or MIN.
- Change Measure Type: Switch the measure type between String and Number.

7.3.3.4.3. Toolbar and shortcut menu

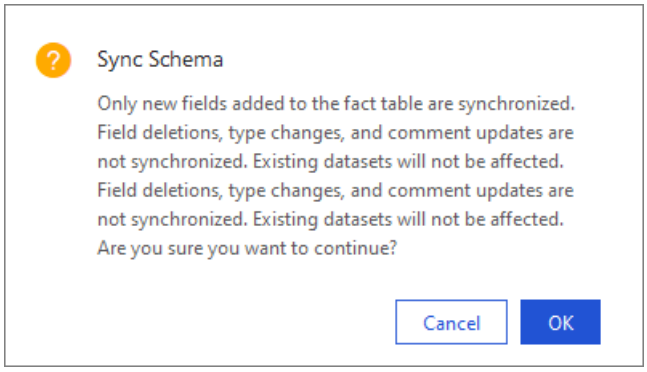
On the dataset edit page, Quick BI provides a toolbar and a shortcut menu for you to edit a dataset and create reports.

Toolbar

- Lock: provides a lock mechanism. Multiple users can edit a dataset at the same time. If you want to modify a dataset, you must first lock the dataset and refresh the page to update the dataset. Then, make and save the modifications. If you modify the dataset without refreshing the page, the changes made by other users are overwritten.
- Sync Schema: synchronizes the new fields in an online physical table to the dataset that is created based on the physical table.

  However, if a field is modified or deleted from the physical table, the field in the dataset remains unchanged.

  Sync Schema

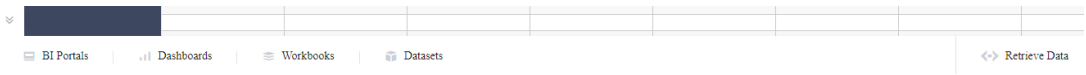
- Refresh Preview: refreshes the dataset for preview. If you want to view the latest data, save the dataset and click Refresh Preview.
- Set Filter: filters out unnecessary data in the dataset to prevent full table scan.
- Save As: saves the current dataset as a new dataset. This function allows you to duplicate or back up a dataset.
- Save: saves the dataset.

  After you add new fields, delete fields, or convert between dimensions and measures in a dataset, you must save the dataset. Then, you can refresh the page to view the updated dataset.

Shortcut menu

On the dataset edit page, click the icon in the lower-left corner to expand the shortcut menu.

Shortcut menu

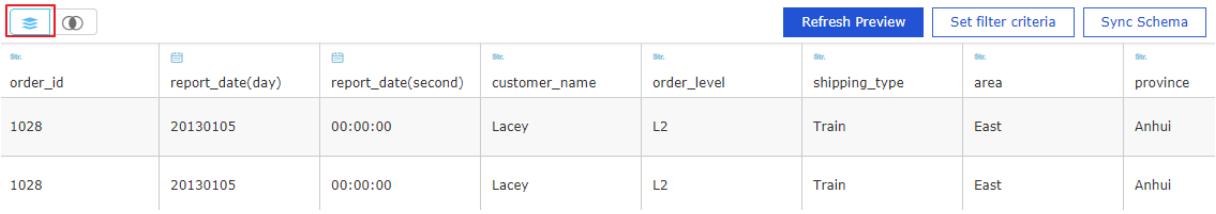


- Dashboards: You can click **Dashboards** to open the **Create Dashboard** dialog box.
- Workbooks: You can click **Workbooks** to go to the page for creating and editing a workbook.
- Datasets: You can click **Datasets** to go to the page for creating a dataset.
- BI Portals: You can click **BI Portals** to go to the page for creating and editing a BI portal.
- Retrieve Data: You can click **Retrieve Data** to go to the page for creating a data source.

7.3.3.4.4. Preview data

This topic describes how to preview data on the dataset edit page.

Click the icon to preview data.



order_id	report_date(day)	report_date(second)	customer_name	order_level	shipping_type	area	province
1028	20130105	00:00:00	Lacey	L2	Train	East	Anhui
1028	20130105	00:00:00	Lacey	L2	Train	East	Anhui

7.3.3.4.5. Join tables

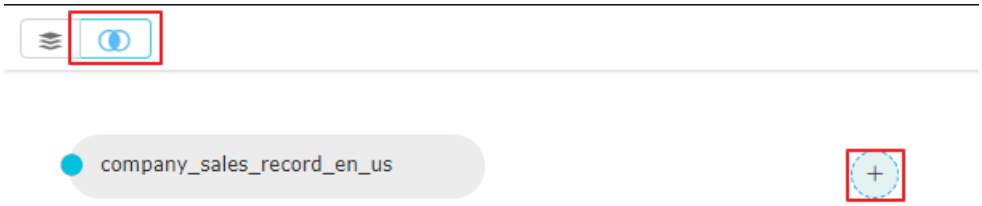
This topic describes how to join tables when you edit a dataset.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the Datasets page, click the target dataset to go to the dataset edit page. The dataset

company_sales_record is used as an example.

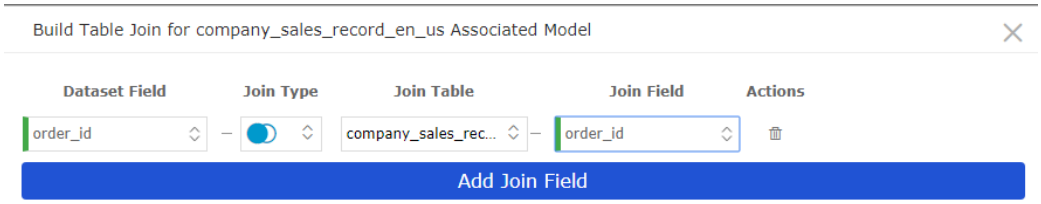
5. Click the **Join** icon ( ) to switch to the Table Join mode.



6. Click the plus sign (+). The **Build Table Join for company_sales_record Associated Model** dialog box appears. Quick BI supports five-layer horizontal join, as shown in the following figure.

7. Select the fields that are used to join tables and specify a join type.

Build a table join model



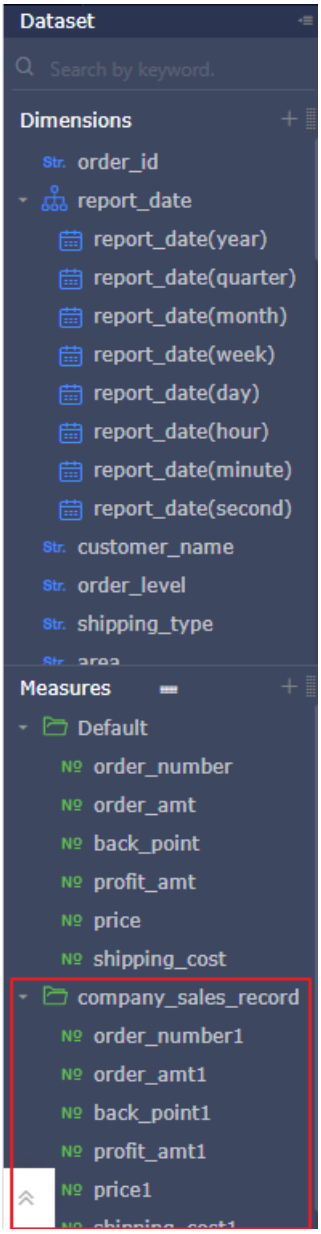
Quick BI supports the following join types:

- Inner Join ( ): returns records with the same join fields in two tables.
- Left Outer Join ( ): returns all records in the left table and records in the right table that have the same join fields as the left table.
- All Join ( ): returns all records in two tables.

> **Note** MySQL data sources do not support the All Join type.

8. Click **Add Join Field** to add multiple join fields.
9. Click **OK** to save the model.

10. Click **Save**.
11. After the model is saved, click the preview icon ( ) to switch to the preview mode, and then click **Refresh Preview** to view the join results of the tables.



7.3.3.4.6. Calculated fields

7.3.3.4.6.1. Overview

Quick BI allows you to create fields from existing fields in data sources by using SQL functions. The new fields are referred to as calculated fields.

When you create a calculated field, take note of the following points:

- Fields that are created from the Dimensions list are classified as calculated dimensions. Fields that are created from the Measures list are classified as calculated measures.

- In the Expression field, you can select any functions or expressions supported by the current data source.
- You must enter the function name. The field name must be enclosed in a pair of brackets. You can enter the entire field name, or enter a left bracket ([) and select the required field from the displayed field list. You can also double-click the target field in the Dimensions or Measures list to add the field to the Expression field. Correct SQL expressions are highlighted.
- When you specify an SQL expression for a calculated field, do not use full-width punctuation marks. If an error occurs, check whether you have entered full-width punctuation marks in the expression field.
- After you create the calculated field, save the dataset and then refresh it to show the calculated field.
- You cannot use a calculated field in the expression of another calculated field. If a field is deleted from the physical table, a calculated field whose expression uses the deleted field becomes invalid.

7.3.3.4.6.2. Rules for using calculated fields

This topic describes the rules for using calculated fields.

- Calculated fields not using aggregate functions can be used as dimensions, or as measures after you specify the aggregate method. Calculated fields using aggregate functions can only be used as measures, and cannot be converted to dimensions.
- You can set the data type for a calculated field. Currently, you can set the data type of dimension fields to Number, Text, or Date/Time, and set the data type of measure fields to Number or Text.
- Similar to dimensions and measures generated by the original fields in a data source, dimensions and measures generated by calculated fields can be used in rows, columns, and filters, and can be selected on the Data tab for charts and maps. You can also convert a calculated field from dimension to measure or from measure to dimension.

> **Note** If you use the SUM or AVG aggregate function in an expression of a calculated field whose data type is Text and values are text data, an error occurs due to a failure in data type conversion.

7.3.3.4.6.3. Types of calculated measures

A calculated field in the measure category is a calculated measure. Calculated measures are classified into common measures and aggregate measures based on the expression type.

The following table lists differences between common measures and aggregate measures.

Common measure	Aggregate measure
Expressions exclude aggregate functions.	Expressions include aggregate functions.
You can change the aggregation method.	You cannot change the aggregation method.
You can convert common measures to dimensions.	You cannot convert aggregate measures to dimensions.

Common measure	Aggregate measure
No aggregate functions are supported.	Supported aggregate functions: SUM, AVG, MIN, MAX, COUNT, and COUNT DISTINCT.

You can use the COUNT or COUNT DISTINCT function that has dimensions as its parameters to form a deduplicated aggregate measure.

For example, to calculate the average purchase price per user, you can use the following expression: sum(purchase price)/countd(user ID). To calculate the proportion of order costs to order prices, you can use sum(order cost)/sum(order price) but not avg(order cost/order price).

> **Note** Common measures cannot be used with aggregate measures. For example, sum(order cost)/order price is incorrect.

7.3.3.4.6.4. Expressions of calculated fields

This topic describes functions and arithmetic operations of calculated fields.

Aggregation methods

- To calculate the total order price: sum([order_amt])
- To calculate the average order price: avg([order_amt])
- To calculate the maximum order price: max([order_amt])
- To calculate the minimum order price: min([order_amt])
- To count the number of customers: count([customer_name])
- To count the number of unique customers: count(distinct [customer_name])

Basic operations

order_cost = ([order_amt] - [profit_amt])/100

Substring

Substring([customer_name],1,1)

Group values of a measure by using the CASE WHEN statement

- Group orders based on the order price

  CASE WHEN [order_amt] < 500 THEN 'small order' WHEN [order_amt] >= 500 AND [order_amt] < 2000 Then 'medium order' WHEN [order_amt] >= 2000 AND [order_amt] < 5000 THEN 'big order' ELSE 'large order' END

- Group dimension members by using the CASE WHEN statement. In this example, provinces are classified into a specific physical region.

  CASE WHEN [province] in ('Heilongjiang', 'Liaoning', 'Jilin') THEN 'Northeast' ELSE [province] END

- Calculate a measure by using a complex expression: order price per customer

  sum([order_amt])/count(distinct[customer_name])

- Add a UNIX timestamp

from_unixtime([order_id] + 1234567890)

- Locate different days in a month

  day([order date])

  Returns a number in the range of 1 to 31.

- Locate different hours in a day

  hour([report_date])

  Returns a number in the range of 0 to 23.

- Calculate the advertisement conversion rate

  CASE WHEN sum([Views]) > 0 THEN sum([Conversion times])/sum([Views]) ELSE 0 END

  The following example is an incorrect expression: sum(CASE WHEN [Views] > 0 THEN [Conversion times]/[Views] ELSE 0 END). For metrics that indicate rates, you must perform the sum operation before the division operation.

7.3.3.4.6.5. Add a calculated field

This topic describes how to add a calculated field.

Prerequisites

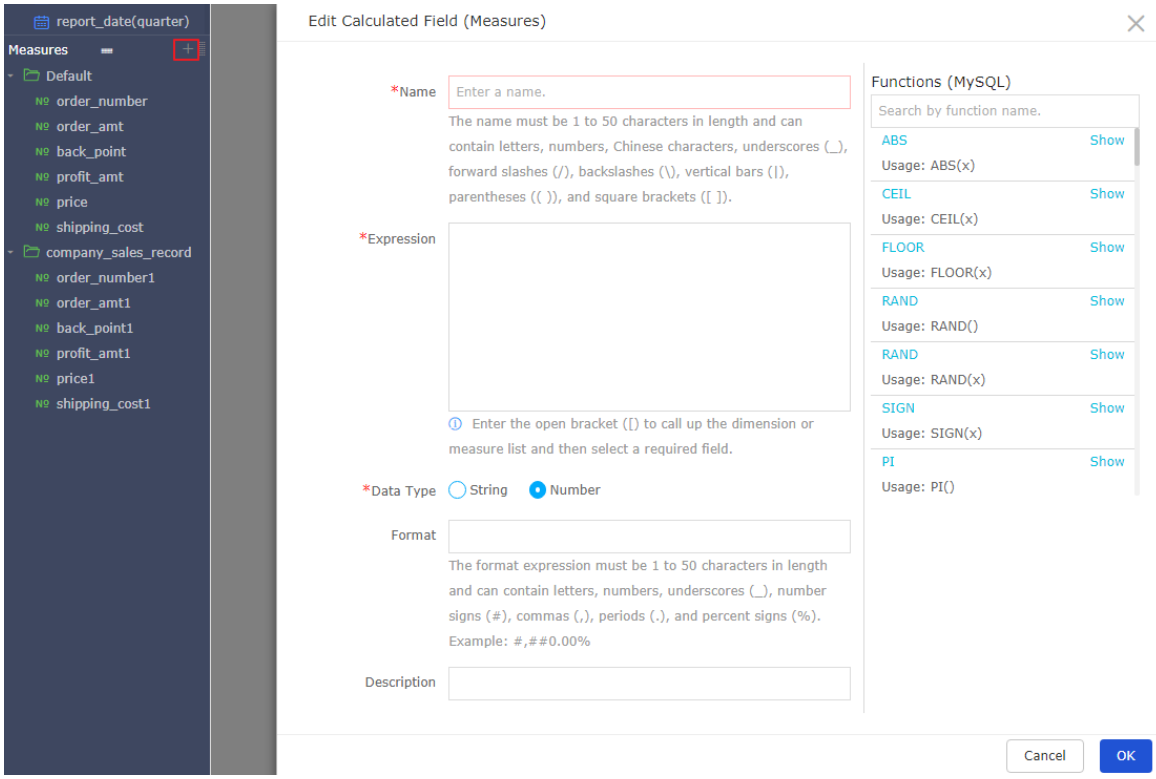
For information about the usage and expressions of calculated fields, see Rules for using calculated fields and Expressions of calculated fields.

Context

The following example uses the company_sales_record dataset to calculate the average profit of orders.

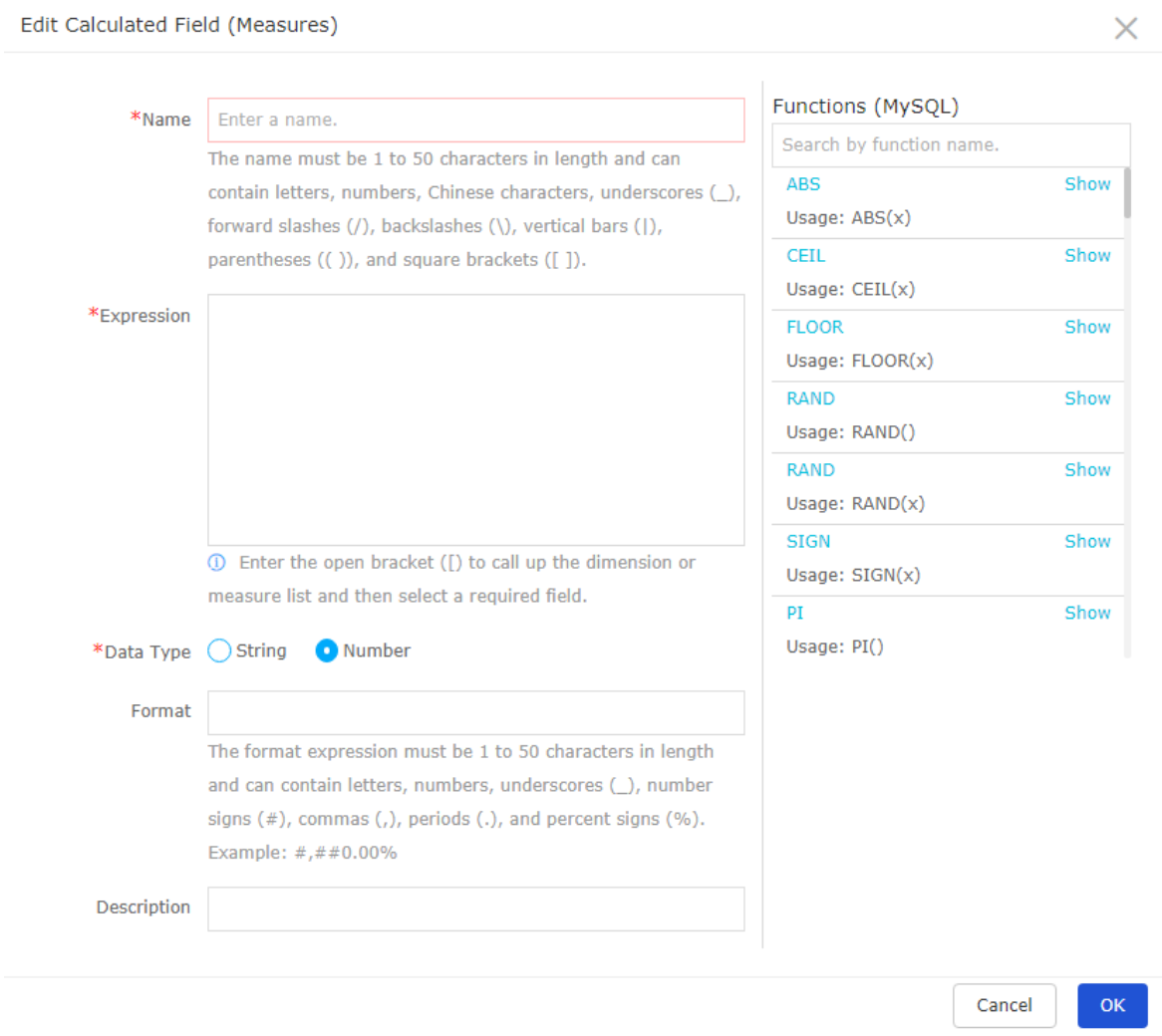
Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Datasets**.
4. On the Datasets page, click the company_sales_record dataset.
5. In the Measures list, click the plus sign (+). The **Edit Calculated Field (Measures)** dialog box appears.



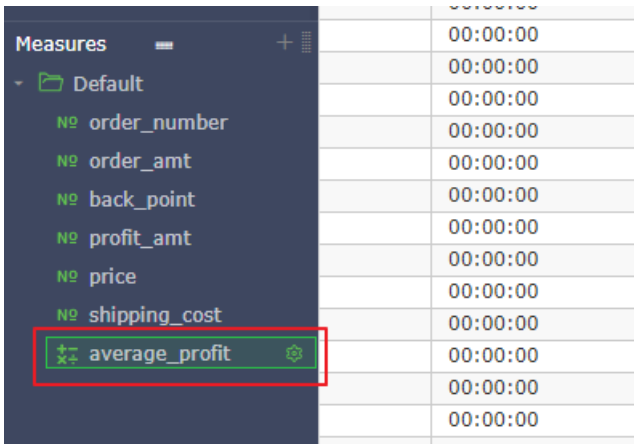
6. Enter the measure name and expression.

   If you want to calculate the average profit of orders, enter an expression to divide the total profit of the orders by the order quantity.



Note You must use an English IME to enter the expression.

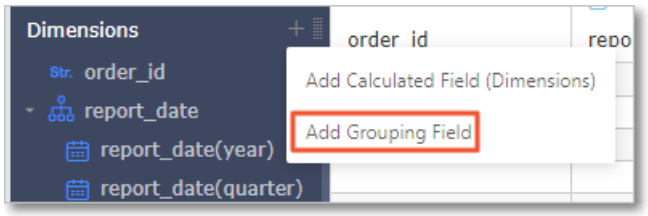
7. Select a data type. For example, if the average profit is a numeric value, you need to select **Number** for Data Type.
8. Click **OK** to add the field.
9. Click **Save** in the upper-right corner to save the dataset.
10. Click **Refresh Preview** to view the new calculated field.



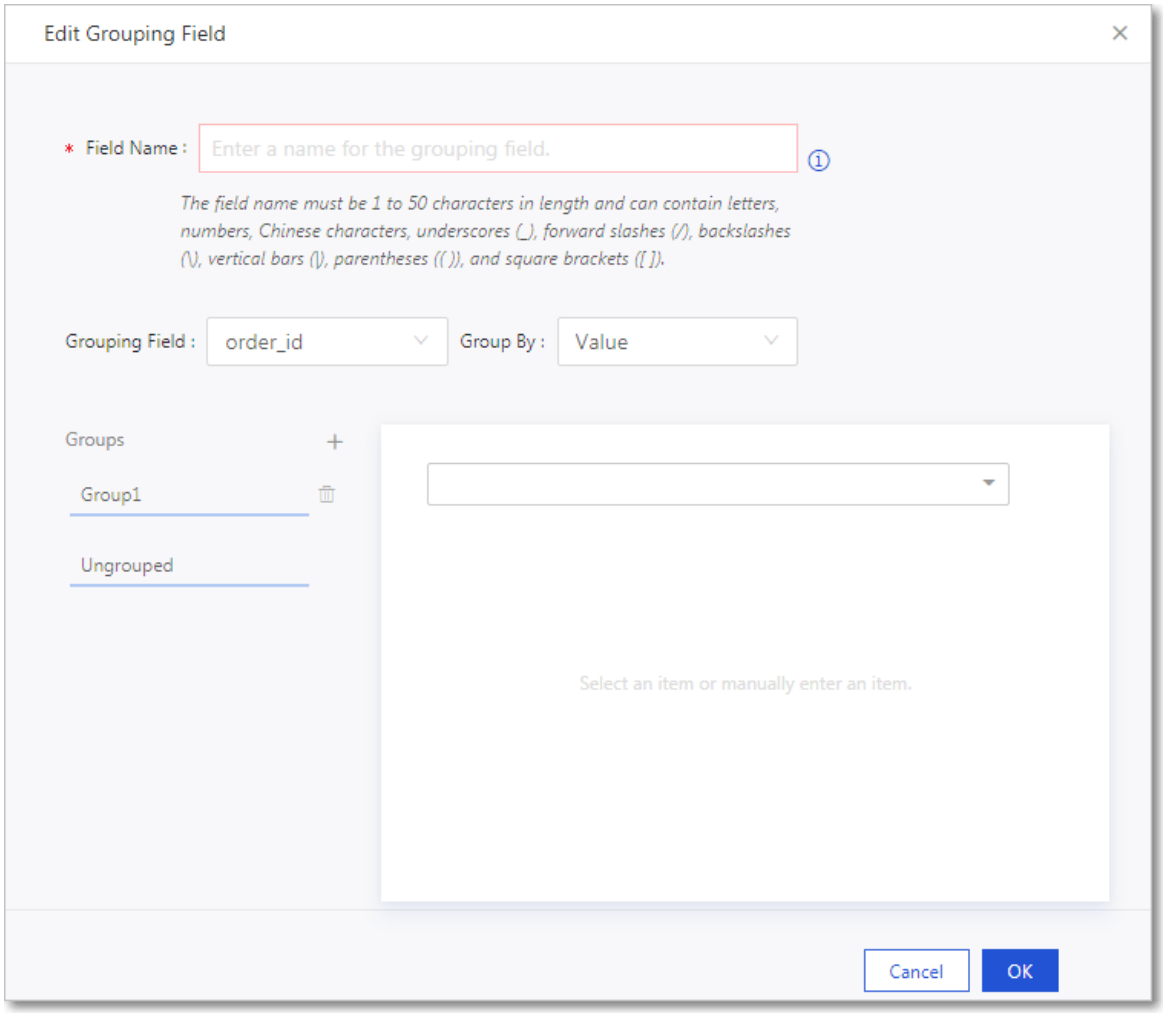
7.3.3.4.7. Add a grouping field

On the dataset edit page, you can use the **Add Grouping Field** feature to group data and store group information.

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Datasets**.
4. On the Datasets page, find the dataset to which you want to add a grouping field, and click the Edit icon in the Actions column. On the dataset edit page, choose **+ > Add Grouping Field** next to Dimensions.



5. In the **Edit Grouping Field** dialog box that appears, enter the required information and click **OK**.



6. Click **Save** and then click **Refresh Preview**. The dimension list shows the new grouping field.

7.3.3.5. Rename a dataset

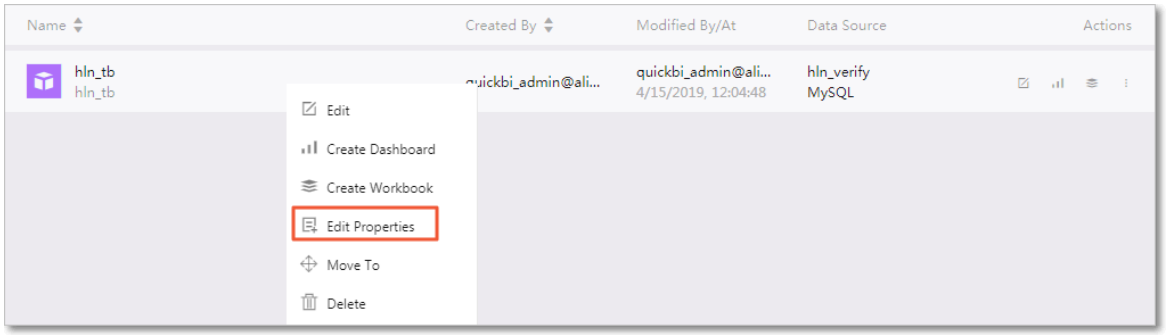
This topic describes how to rename a dataset.

Prerequisites

A dataset is created.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the **Datasets** page, find the dataset that you want to rename. Right-click the dataset or click the **More** icon in the Actions column.
5. Select **Edit Properties**. In the Edit Properties pane, enter a new name for the dataset.



7.3.3.6. Search for a dataset

This topic describes how to search for a dataset.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the **Datasets** page, find the search box.
5. Enter a keyword of the target dataset in the search box and click the icon.

7.3.3.7. Transfer a dataset

This topic describes how to transfer a dataset to another user.

Prerequisites

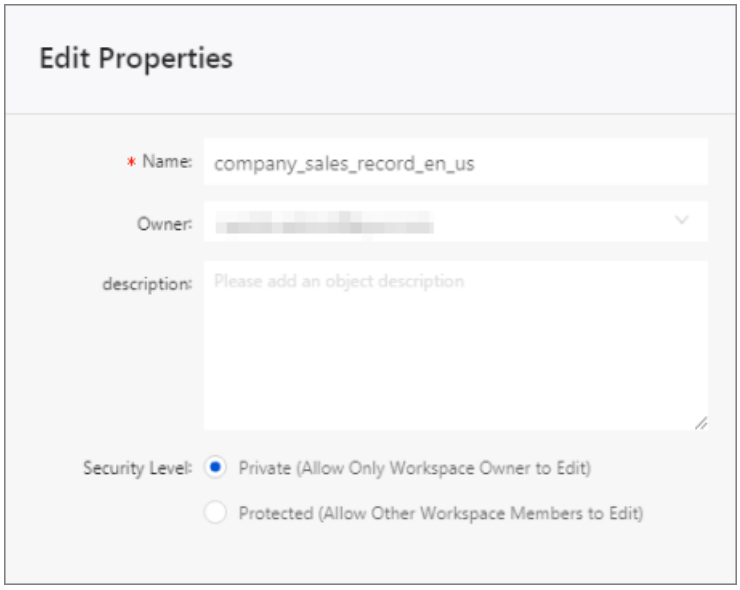
The Quick BI service is purchased.

Context

You can transfer a dataset to another Apsara Stack tenant account or RAM user in the same workspace.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the **Datasets** page, right-click the dataset you want to transfer or click the **More** icon in the Actions column.
5. Select **Edit Properties**.
6. In the **Edit Properties** pane, select a new owner and click **Save**.



7.3.3.8. Copy a dataset from one workspace to another

This topic describes how to copy a dataset from one workspace to another.

Prerequisites

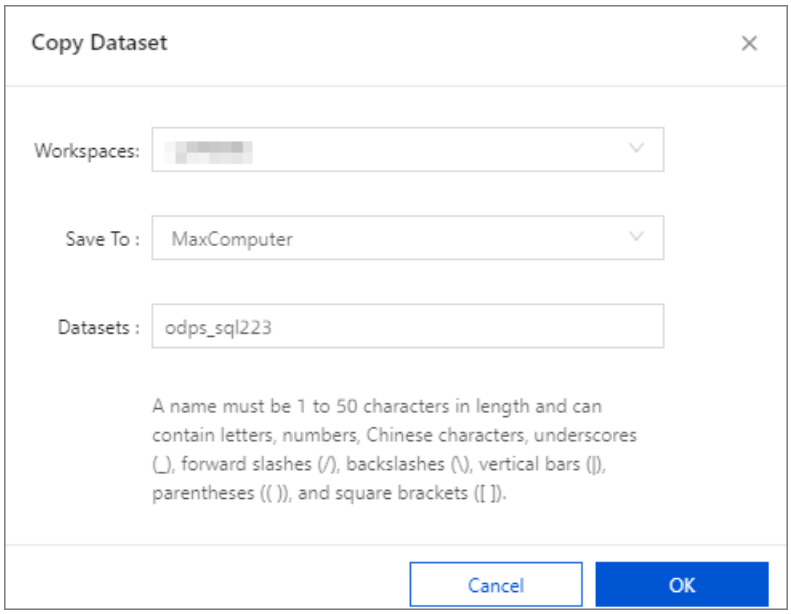
The Quick BI service is purchased.

Context

Only group workspaces support this feature. Only an administrator of two workspaces can copy a dataset from one workspace to the other.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the **Datasets** page, right-click the dataset that you want to copy and select **Copy**.
5. In the **Copy Dataset** dialog box, configure the destination workspace, storage path, and the name of the dataset in the destination workspace.



> **Note** If the destination workspace does not contain the data source based on which the dataset is created, the data source is also copied to the destination workspace.

6. Click **OK**.

7.3.3.9. Create a dataset folder

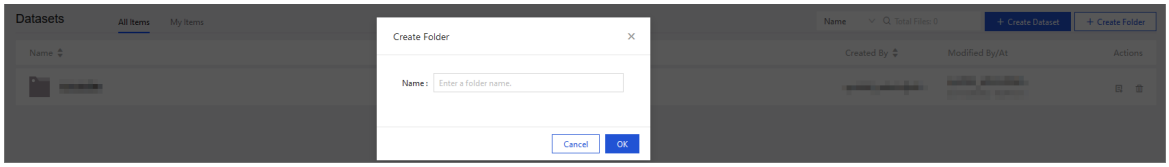
This topic describes how to create a dataset folder on the **Datasets** page.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. In the upper-right corner of the **Datasets** page, click **Create Folder**.
5. In the **Create Folder** dialog box, enter a name for the folder and click **OK**.



7.3.3.10. Rename a dataset folder

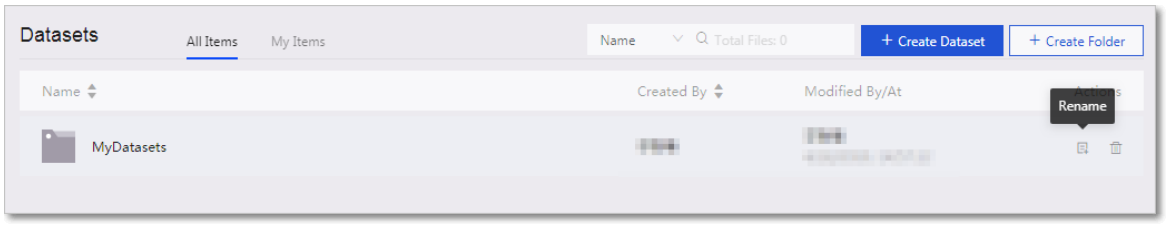
This topic describes how to rename a dataset folder.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane, click **Datasets**.
4. On the **All Items** tab of the **Datasets** page, find the folder that you want to rename.
5. Right-click the folder and select **Rename**, or click the Rename icon in the Actions column.



6. In the Rename dialog box, enter a new name and click **OK**.

7.3.3.11. Delete a dataset

This topic describes how to delete a dataset and the common issues that may occur during this process.

Prerequisites

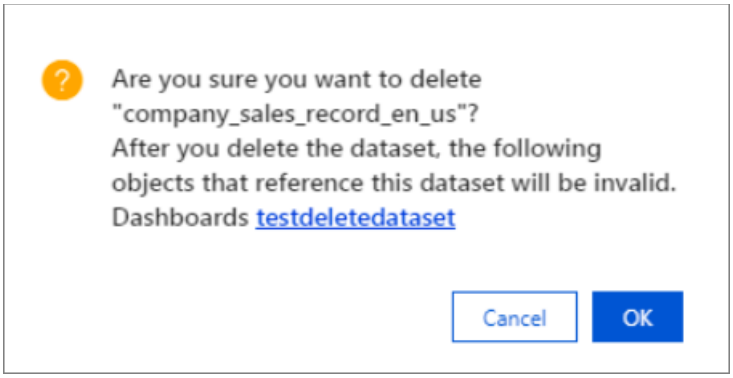
The Quick BI service is purchased.

Context

If workbooks are created based on the dataset, a notification is displayed when you attempt to delete the dataset, as shown in Notification.

After you delete a dataset, an error occurs when you access the dashboard created based on that dataset, as shown in Error message.

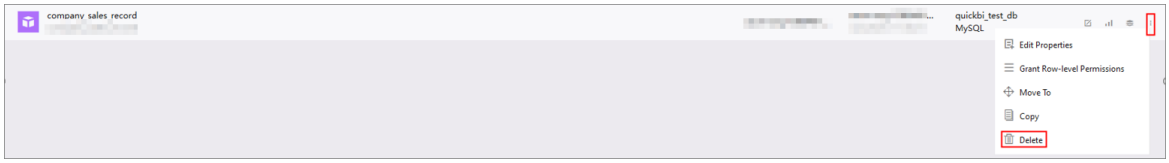
Notification



Error message

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Datasets**.
4. On the **Datasets** page, find the dataset you want to delete.
5. Right-click the dataset and select Delete, or click the **More** icon and select **Delete**.



7.3.3.12. Configure row-level permissions

Row-level permissions are granted based on datasets. Quick BI supports two authorization modes: user/user group-based authorization and tag-based authorization.

User/user group-based authorization is suitable for scenarios that involve a small number of members. Tag-based authorization is suitable for scenarios that involve a large number of members. Tag-based authorization allows you to authorize all users at the same time. In scenarios that involve a large number of members, this mode reduces the costs and complexity of row-level permission configurations and facilitates permission management. For information about how to configure row-level permissions, see Manage row-level permissions.

7.4. Dashboards

7.4.1. Dashboard overview

7.4.1.1. Dashboard features

This topic describes the features of a Quick BI dashboard.

- Supports the Filter Bar, Text Area, IFrame, Tab, and Image widgets. You can drag and drop widgets to create pages for various products.

- Provides a wide range of chart components. You can easily create various reports by setting chart elements. A chart can be displayed in either standard or full-screen mode.
- Uses a more flexible tile layout. A dashboard visualizes data. It allows you to filter and query data, use multiple data display modes, and highlight the key fields of data.
- Enables you to drag, drop, and click fields in a dashboard to display data. You can follow the instructions on the pages to analyze data for better user experience.

7.4.1.2. Chart types and scenarios

Different types of data need to be displayed in charts of different types. Currently, Quick BI supports various data charts, such as line charts, vertical bar charts, bubble maps, and funnel charts.

For information about how to create charts, see Create a dashboard.

The following table describes the analysis types and scenarios for each chart type.

Chart types and scenarios

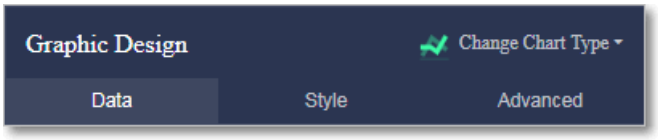
Analysis type	Description	Scenario	Applicable chart
Comparison	Compares values or compares measures by category	Compares the sales or income between different countries or regions.	Vertical bar chart, radar chart, funnel chart, cross table, polar diagram, tornado-leaned funnel chart, and word cloud
Percentage	Displays the proportion of a part to the total, or the proportion of a certain value to the whole.	Displays the sales of the salesperson that has the greatest percentage of total sales.	Pie chart, funnel chart, gauge, and treemap
Relationship	Displays the relationship between values or measures.	Displays the relationship between two measures. This helps you understand the influence the first measure has on the second measure.	Scatter chart, treemap, kanban, hierarchy chart, and flow analysis chart
Trend	Displays data trends, especially trends based on time, such as year, month, or day, or the progress of a data metric or other possible patterns.	Displays trends in sales or revenue for a product over a period of time.	Line chart

Analysis type	Description	Scenario	Applicable chart
Geographic map	Displays the size and distribution of data metrics of a country or region on a map. The datasets must contain geographic data.	Displays the income information about different regions of a country.	Bubble map and colored map

7.4.1.3. Data elements of a chart

This topic describes the data elements of a chart.

The graphic design for each chart has three tabs: **Data**, **Style**, and **Advanced**.



- On the **Data** tab, you can specify the data you want to display in the chart.
- On the **Style** tab, you can set the chart layout and details of the chart.
- On the **Advanced** tab, you can configure the filter interaction feature to dynamically compare and display data.

Each chart has unique core data elements. For example, in a geo map, a latitude field is required. Otherwise, data cannot be displayed.

The following table lists the core data elements of each chart type.

Data elements of different types of charts

Chart type	Required data element	Data element description
Line chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Stacked line chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Area chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Stacked area chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.

Chart type	Required data element	Data element description
100% stacked area chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Vertical bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Stacked vertical bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
100% stacked vertical bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Circular bar	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Horizontal bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Stacked horizontal bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
100% stacked horizontal bar chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Combination chart	Category axis and value axis	You must specify at least one dimension for the category axis and at least one measure for the value axis.
Pie chart	Label and central angle	You can specify only one dimension for labels and only one measure for the central angle.

Chart type	Required data element	Data element description
Bubble map	Geo location and bubble size	You can specify only one dimension for geo locations. The dimension type must be Geo. You can specify one to five measures for the bubble size.
Colored map	Geo location and colorscale	You can specify only one dimension for geo locations. The dimension type must be Geo. You can specify one to five measures for the colorscale.
Geo map	Geo location and colorscale	You can specify only one dimension for geo locations. The dimension type must be Geo. You can specify one to five measures for the colorscale.
Geo bubble map	Geo location and colorscale	You can specify only one dimension for geo locations. The dimension type must be Geo. You can specify one to five measures for the colorscale.
Cross table	Rows and columns	You can specify an unlimited number of dimensions for the rows and an unlimited number of measures for the columns.
Pivot table	Rows and values	You can specify an unlimited number of dimensions for the rows and an unlimited number of measures for the columns.
Gauge	Pointer angle	You can specify only one measure for the pointer angle.
Progress bar	Pointer	You must specify one to five measures for the pointer.
Radar chart	Label and length	You must specify one or two dimensions for labels and at least one measure for the length.

Chart type	Required data element	Data element description
Scatter chart	Color legend, x-axis, and y-axis	You can specify only one dimension for the color legend. The maximum number of dimension values is 1,000. You can specify one to three measures for the x-axis and only one measure for the y-axis.
Bubble chart	x-axis, y-axis, and bubble size	You must specify at least one dimension for the x-axis. The number of dimension values is up to 1,000. You can specify only one measure for the y-axis and only one measure for the bubble size.
Funnel chart	Tier label and tier area	You can specify only one dimension for tier labels and only one measure for tier areas.
Kanban	Label and metric	You can specify only one dimension for labels and one to ten measures for metrics.
Treemap	Label and size	You can specify only one dimension for labels and only one measure for the size.
Polar diagram	Arc radius and label	You can specify only one dimension for the label and only one measure for the arc radius.
Word cloud	Word size and word	You can specify only one dimension for word sizes and only one measure for words.
Tornado-leaned funnel chart	Metrics for measures and dimensions	You can specify one dimension and one measure for data comparison.
Hierarchy chart	Node label and node metric	You must specify at least two dimensions for node labels and at least one measure for node metrics.
Flow analysis chart	Central node, node type, node name, and node indicator	You can specify only one dimension or one measure for each data element.

Chart type	Required data element	Data element description
Waterfall chart	Category axis and value axis	You must specify at least one dimension for the category axis and only one measure for the value axis.
Trend indicator	Date and indicator	You can specify only one dimension for the date and at least one measure for the indicator.
Sankey diagram	Node type and node height	You can specify two to five dimensions for the node type and only one measure for the node height.
Ranking board	Category and indicator	You can specify only one dimension for the category and only one measure for the indicator.
Ticker board	Indicator	You can specify only one dimension for the indicator.

7.4.2. Access a dashboard

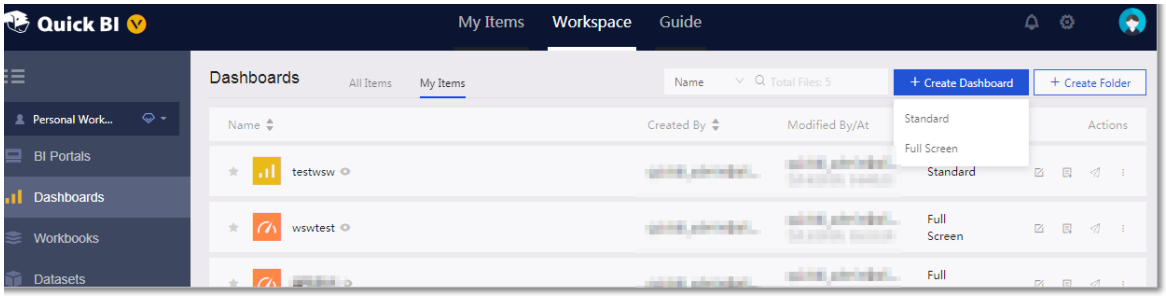
This topic describes how to access a dashboard.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. Click the **Workspace** tab.
3. In the left-side navigation pane of the Workspace page, click **Dashboards**.
4. Choose **Create Dashboard** > **Standard** to go to the dashboard.



7.4.3. Areas of a dashboard

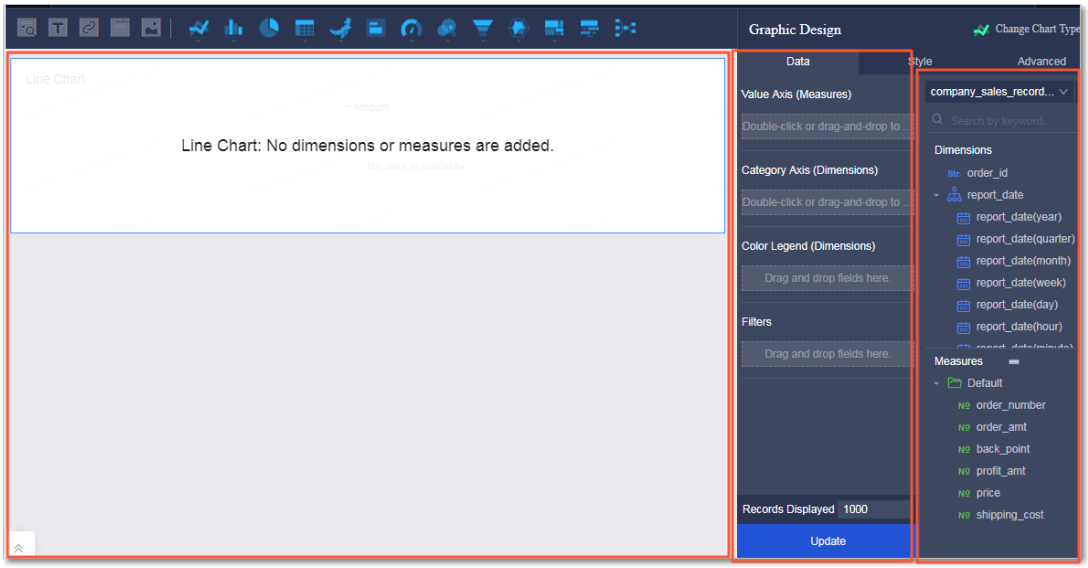
7.4.3.1. Overview

This topic describes the functional areas of a dashboard.

The dashboard edit page contains three areas, as shown in Dashboard.

- Dataset selection area
- Dashboard configuration area
- Dashboard display area

Dashboard



- Dataset selection area: In this area, you can switch the current dataset to another. The fields of each dataset are displayed in the respective Dimensions and Measures lists based on the data types preset in the system. You can select dimensions and measures based on the data elements in the chart.
- In the dashboard configuration area, you can select a chart type, and edit the title, layout, and legend position of a chart. On the Advanced tab, you can associate the current chart with other charts and display analysis results from multiple perspectives. You can also filter data by using filters, or insert a Filter Bar widget to query key data in a chart.
- Dashboard display area: In this area, you can drag charts to adjust their positions and change chart types. For example, you can change a bar chart to a bubble map. The system displays information about missing or error elements. In the dashboard display area, you can save, preview, or create a dashboard. The dashboard provides instructions to help you learn how to create a dashboard.

7.4.3.2. Dataset selection area

7.4.3.2.1. Switch datasets

This topic describes how to switch datasets in a dashboard.

Prerequisites

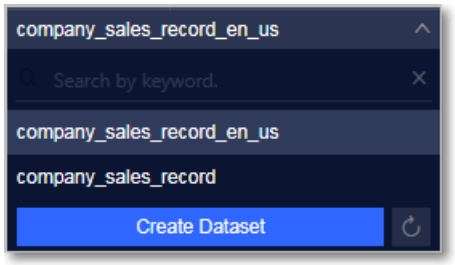
The Quick BI service is purchased.

Context

If you cannot find the required dataset in the drop-down list, go back to the dataset management page and ensure that you created the dataset. For information about how to create a dataset, see Create a dataset.

Procedure

1. Go to the target dashboard.
2. On the Data tab, click the **Switch Datasets** icon.
3. Select the target dataset from the drop-down list that appears.



7.4.3.2.2. Search for a dimension or measure

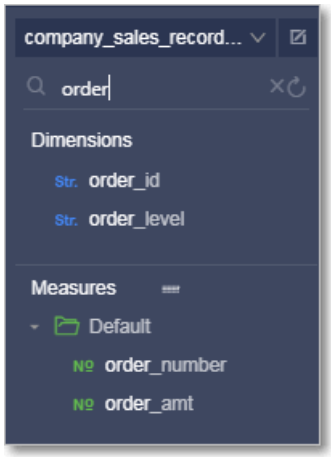
This topic describes how to search for a dimension or measure.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Enter a keyword of a field, such as **order**, in the search box.
3. Click the **Search** icon.



For information about how to edit dimensions and measures, see Edit a dimension and Edit a measure.

7.4.3.3. Dashboard graphic design area

7.4.3.3.1. Select fields

This topic describes how to select fields from a dataset.

Prerequisites

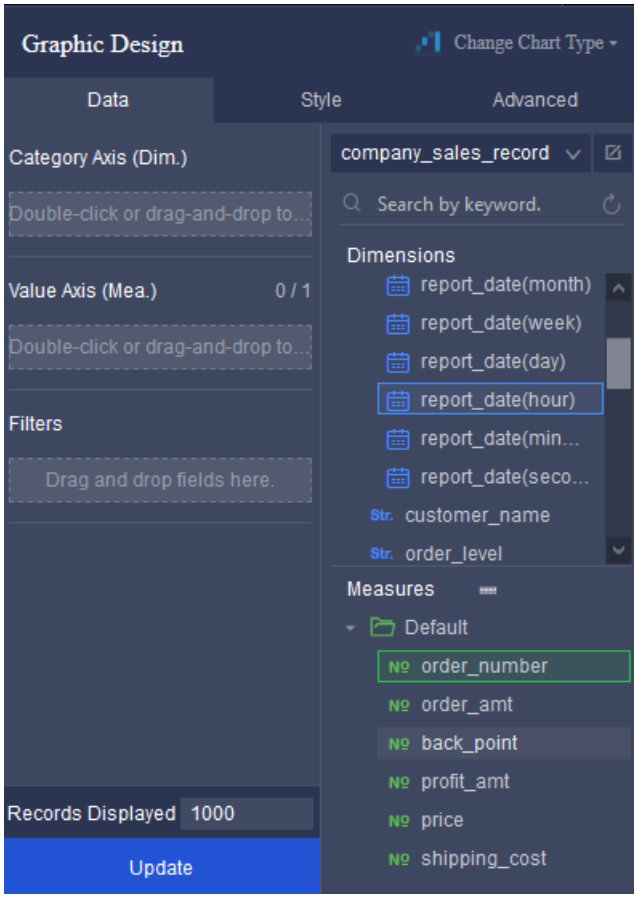
- The Quick BI service is purchased.
- A dataset is selected in the dataset selection area and is edited as required. For information about how to edit a dataset, see Edit a dataset.

Context

Procedure

1. Go to the target dashboard.
2. Select a chart from the toolbar on the top of the dashboard.
3. Click the chart icon. The chart appears in the display area of the dashboard. If you want to change the chart type, click **Change Chart Type** in the Graphic Design area, and select the desired chart type.
4. On the **Data** tab, select the required fields, as shown in Select fields. Double-click a field to add it to the Dimensions or Measures list, or drag the field to the target list.

Select fields



- If you want to delete a field, click the Delete icon next to the field.
- If you want to sort the values of a field, click the ascending or descending icon next to the field.
- You can click the aggregation method icon next to the field and select the method as required. The aggregation methods include SUM, COUNT, MAX, MIN, AVG, COUNT DISTINCT, median, percentile, variance, standard deviation, and compare operations.

5. Click **Update**. The chart is updated.

7.4.3.3.2. Use the color legend

This topic describes how to use the color legend.

Prerequisites

- The Quick BI service is purchased.
- A dataset is selected in the dataset selection area and is edited as required. For information about how to edit a dataset, see Edit a dataset.

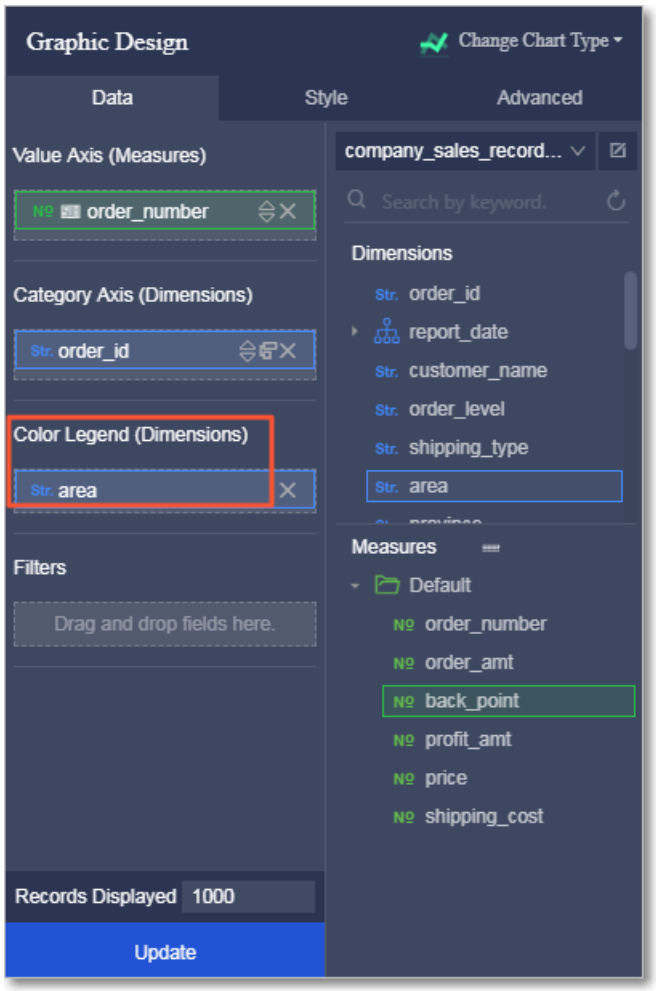
Context

The color legend feature displays the values of a selected field in different colors in a chart.

Only dimensions can be added to the Color Legend field.

Procedure

1. Go to the target dashboard.
2. Drag a dimension, such as product_type, to the **Color Legend (Dimensions)** field.



3. Click **Update**. The values of the field are displayed in different colors in the chart.

4. Change the legend color in the **Series Settings** section on the **Style** tab.

7.4.3.3.3. Sort field data

This topic describes how to sort data on the **Data** tab.

Prerequisites

- The Quick BI service is purchased.
- A dataset is selected in the dataset selection area and is edited as needed. For information about how to edit a dataset, see Edit a dataset.

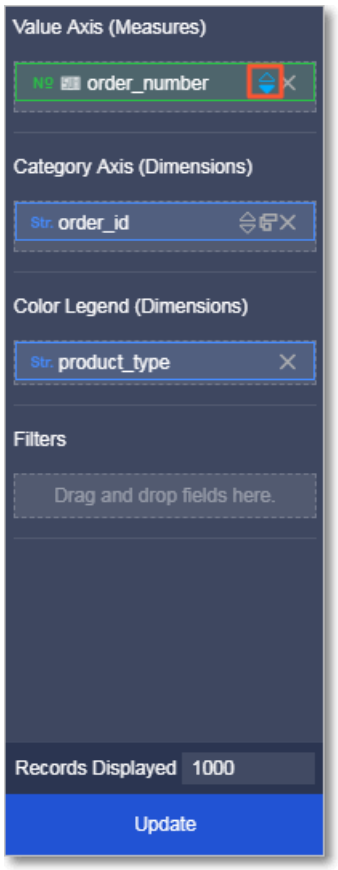
Procedure

1. Go to the target dashboard.

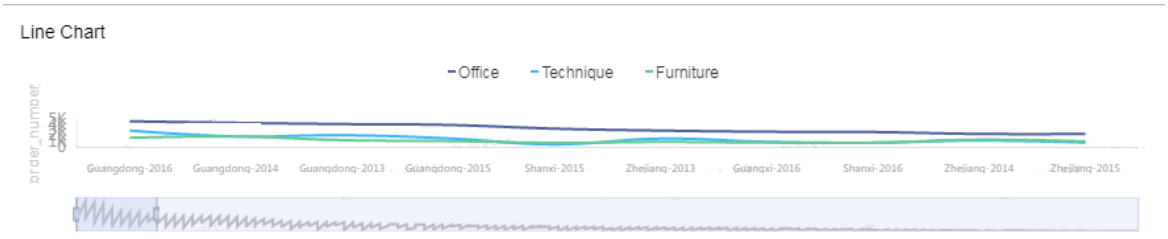
2. Select a field, such as order_number.
3. Click the triangle icon next to the field, as shown in Set the sorting order.

   The upward arrow indicates ascending order, and the downward arrow indicates descending order.

   Set the sorting order



4. After you specify the sorting order, click **Update**.



7.4.3.3.4. Filter by field

This topic describes how to filter data based on specified fields.

Context

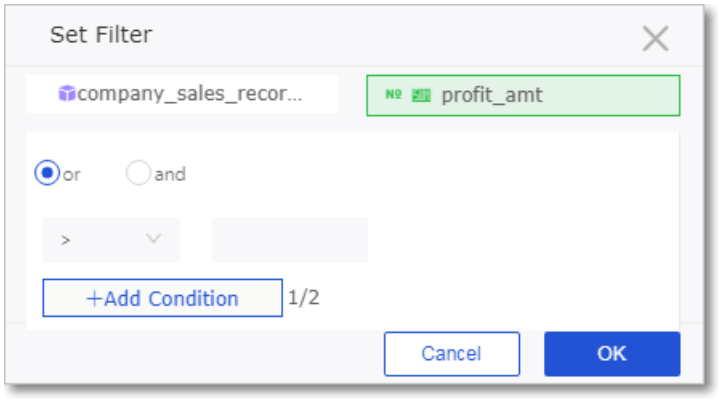
Drag a dimension or measure to the **Filters** area to specify the fields used to filter data.

In the following example, the filed profit_amt is selected.

Procedure

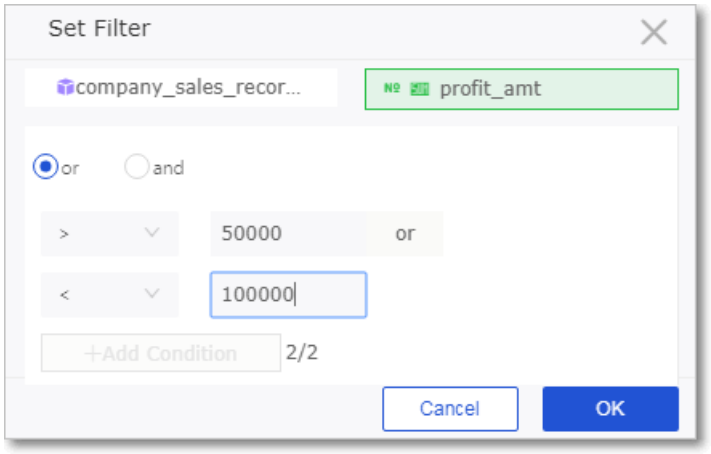
1. Go to the target dashboard.
2. Drag the profit_amt field to the Filters area.
3. Click the **Filter** icon and set the parameters in the Set Filter dialog box, as shown in Set the filter.

   Set the filter



4. Select the filter condition, for example, >, <, or =, as shown in Specify a value range.

   Specify a value range



5. After you set the parameters, click **OK**.
6. Click **Update**. The system then updates the chart based on the parameters of the filter.

7.4.3.3.5. Filter interaction

You can use the filter interaction feature when you have created multiple charts on a dashboard. This topic describes how to use the filter interaction feature.

Context

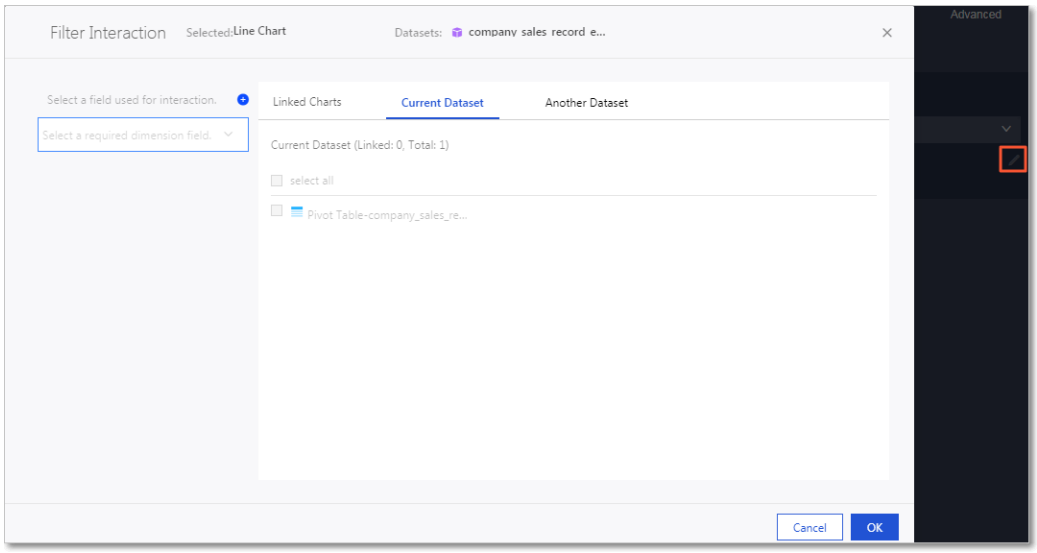
You can configure the filter interaction feature on the **Advanced** tab page.

Before you use this feature, make sure that you have created at least two charts in the current dashboard.

Procedure

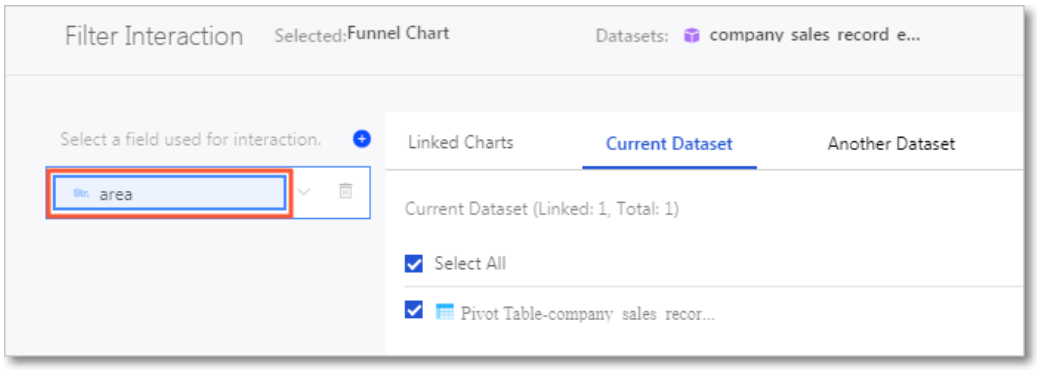
1. Go to the target dashboard.
2. Select a chart, for example, a funnel chart.
3. In the Graphic Design area, click the **Advanced** tab.
4. On the **Advanced** tab page, click the Filter Interaction icon. The system then displays all available charts, as shown in The Advanced tab.

   The Advanced tab



5. Select fields the same as the filter fields from the available charts to associate these charts, as shown in Filter interaction settings.

   Filter interaction settings

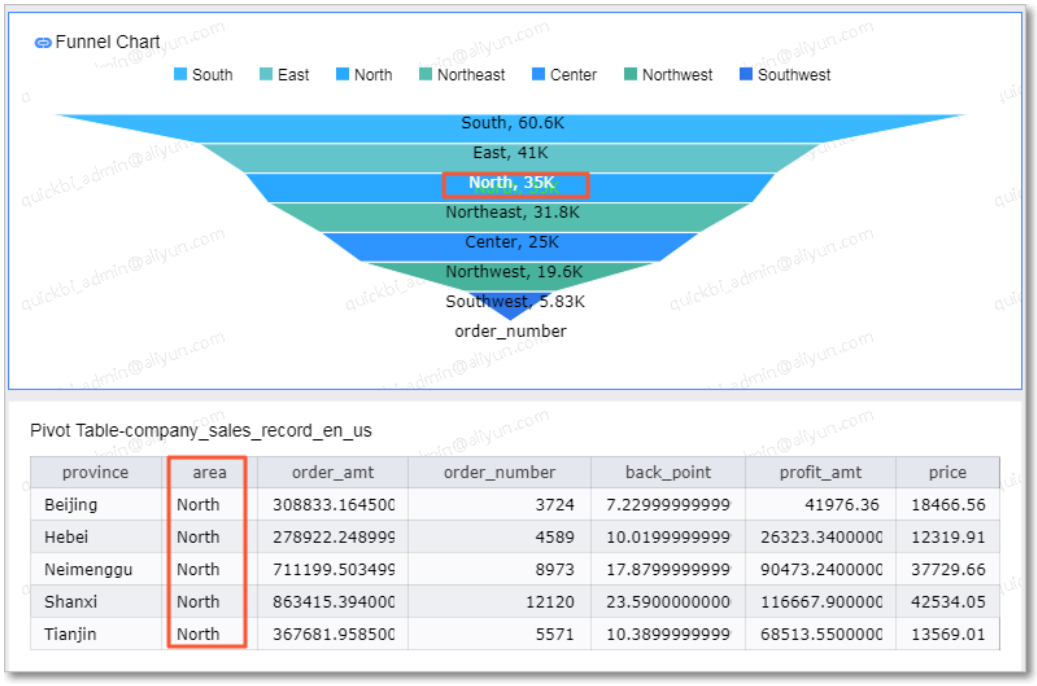


6. In the upper-right corner of the dashboard, click **Preview** to preview the current dashboard.

   Click Preview

7. Click **China North** in the funnel chart, and the associated cross tab chart displays data of China North, as shown in Result.

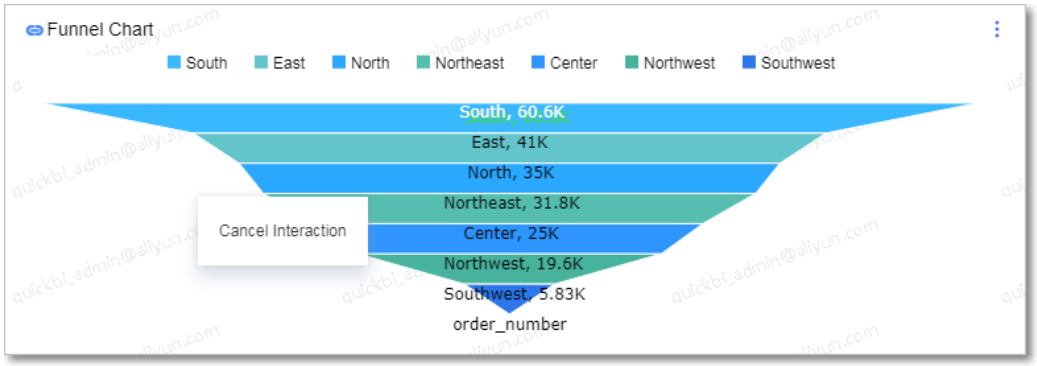
   Result



province	area	order_amt	order_number	back_point	profit_amt	price
Beijing	North	308833.164500	3724	7.22999999999	41976.36	18466.56
Hebei	North	278922.248999	4589	10.0199999999	26323.340000	12319.91
Neimenggu	North	711199.503499	8973	17.8799999999	90473.240000	37729.66
Shanxi	North	863415.394000	12120	23.5900000000	116667.90000	42534.05
Tianjin	North	367681.958500	5571	10.3899999999	68513.550000	13569.01

8. Hover over any blank area in the funnel chart, and click the **Cancel Interaction** notification to disable the filter interaction feature.

   Click the notification



7.4.3.3.6. Metric analysis

This topic describes four methods of analysis: auxiliary line, trendline, prediction, and anomaly detection.

Prerequisites

- The Quick BI service is purchased.
- A dataset is selected in the dataset selection area and is edited as required.
- Go to the target dashboard.

Background

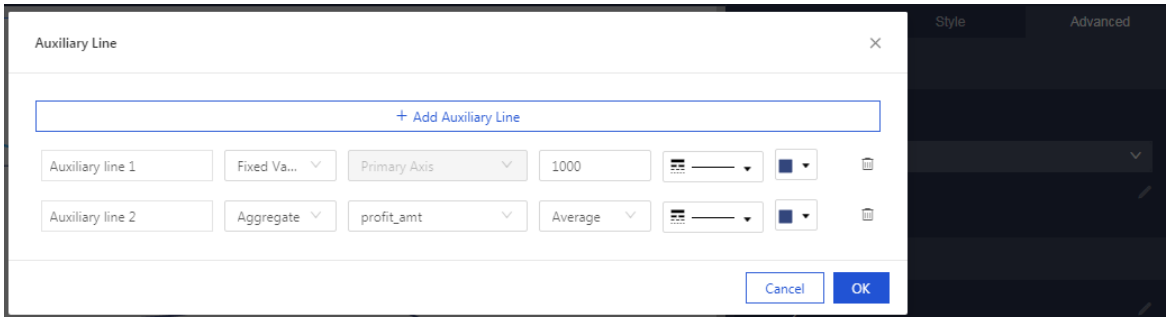
- Metric analysis allows you to analyze data from multiple perspectives. You can use this feature to learn about data trends and anomalies.
- Metric analysis supports four methods of analysis: auxiliary line, trendline, prediction, and

anomaly detection.

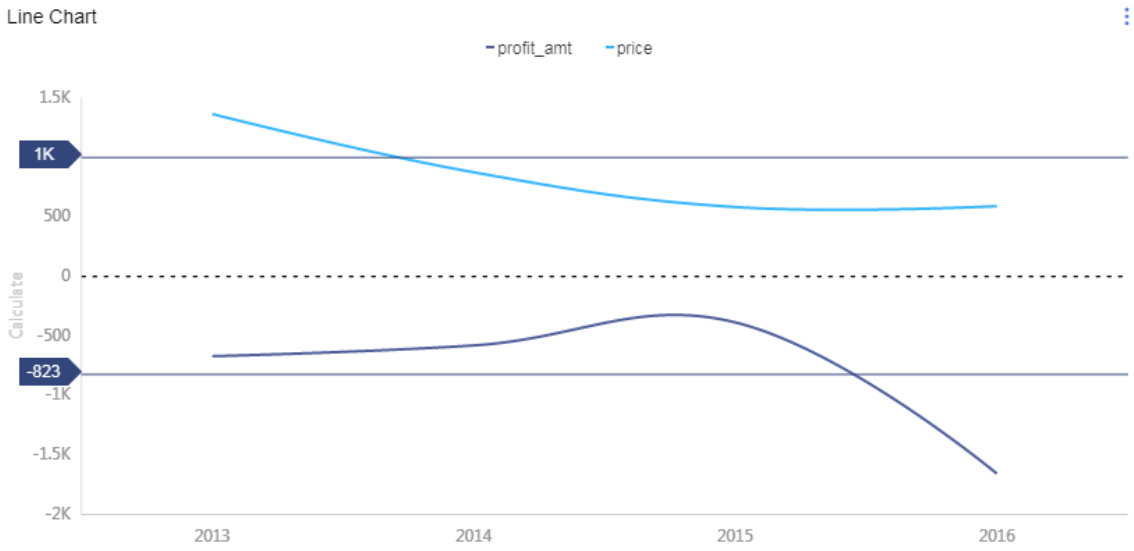
Auxiliary line

You can add an auxiliary line to view the difference between the value of a measure and the value shown by the auxiliary line. The value shown by an auxiliary line is either a fixed value or an aggregate value. Aggregate values includes average, maximum, minimum, and median values.

1. On the **Advanced** tab of the **Graphic Design** page, click next to **Auxiliary line** in the **Metric Analysis** section.
2. In the **Auxiliary Line** dialog box, click Add Auxiliary Line. Select a value type for the auxiliary line you want to create.



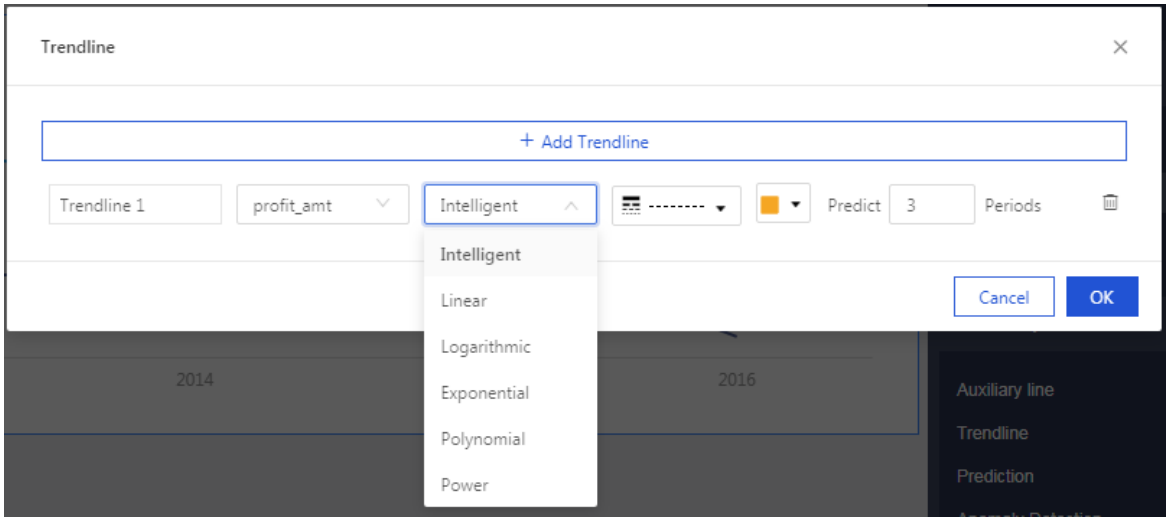
3. Click **OK**. The following figure shows a sample trendline.



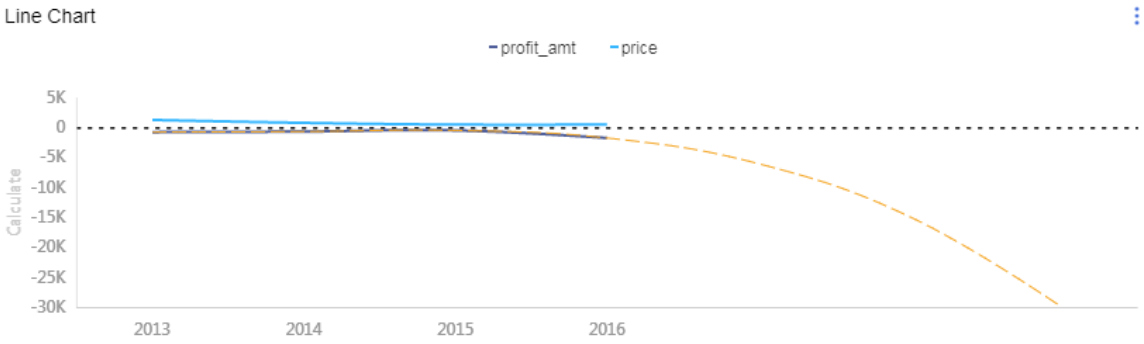
Trendline

A trendline displays data trends. Types of trendlines include Intelligent, Linear, Logarithmic, Exponential, Polynomial, and Power.

1. On the **Advanced** tab of the **Graphic Design** page, click next to **Trendline** in the **Metric Analysis** section.
2. In the **Trendline** dialog box, click Add Trendline. Select a measure, a trendline type, and the number of subsequent periods for which to predict trends.



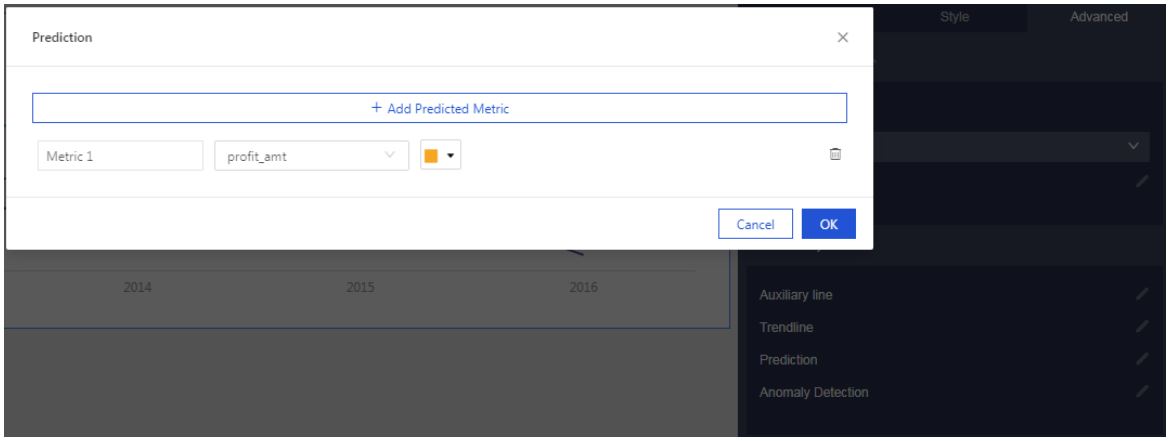
3. Click **OK**. The following figure shows a sample trendline.



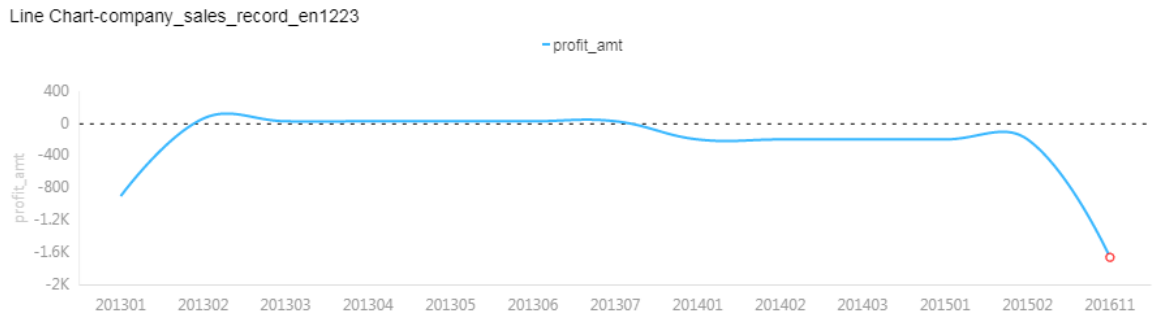
Prediction

You can add a predicted metric to view the trend of current data and predict future trends.

1. On the **Advanced** tab of the **Graphic Design** page, click next to **Prediction** in the **Metric Analysis** section.
2. In the **Prediction** dialog box, click Add Predicted Metric. Select a measure and a color for the line.



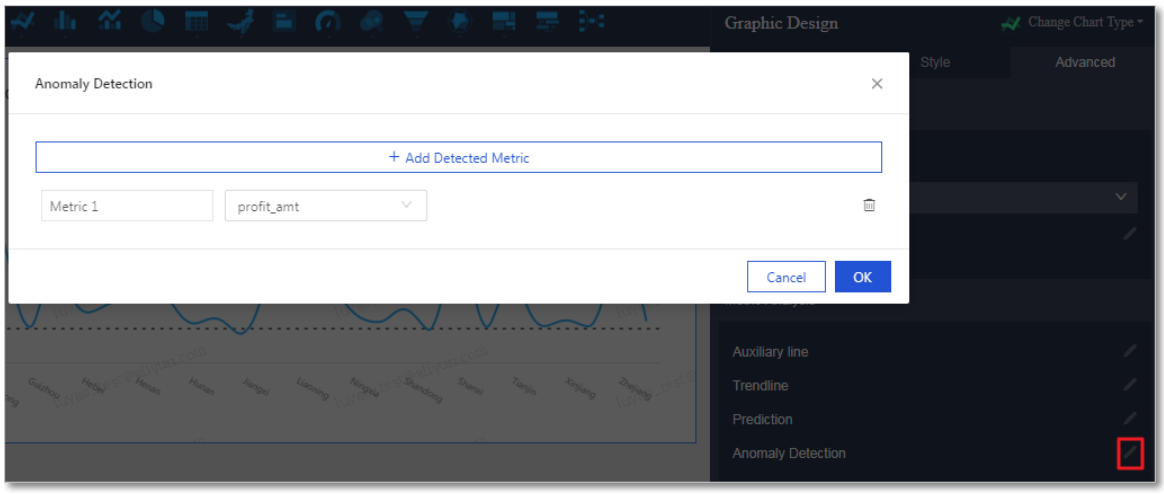
3. Click **OK**. The following figure shows sample prediction results.



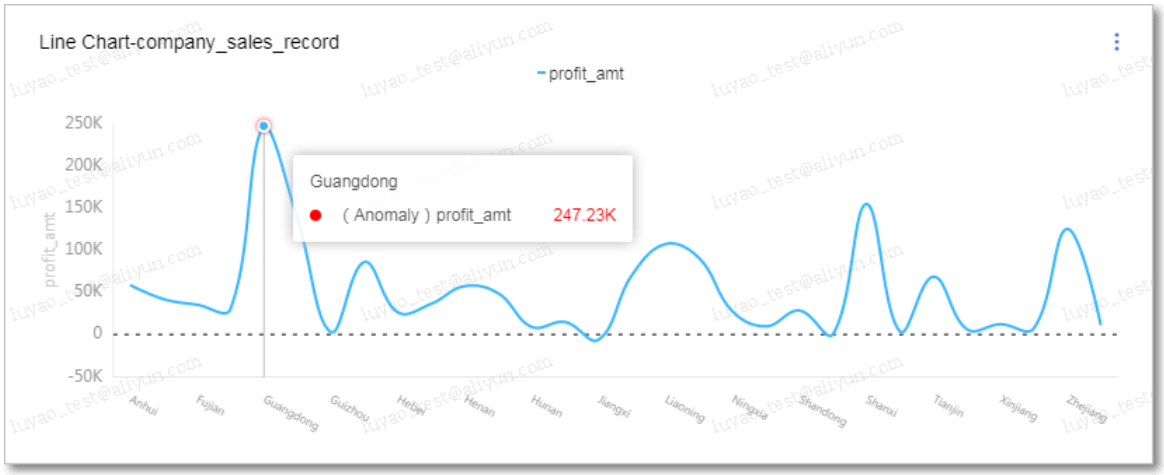
Anomaly Detection

You can add an anomaly detection metric to detect data anomalies.

1. On the **Advanced** tab of the **Graphic Design** page, click next to **Anomaly Detection** in the **Metric Analysis** section.
2. In the **Anomaly Detection** dialog box, click Add Detected Metric. Select a measure.



3. Click **OK**. The following figure shows sample anomaly detection results.



> **Note** In a line chart, anomalies are represented as red dots. In a bar chart, anomalies are represented as red vertical bars.

7.4.3.4. Dashboard display area

7.4.3.4.1. Overview

This topic describes the features of the dashboard display area.

In the display area of a dashboard, you can perform the following operations on one or more charts:

- Adjust chart positions
- View chart data
- Change chart types
- Add to favorites
- Delete a chart

7.4.3.4.2. Toolbar

This topic describes the features of the toolbar.

The toolbar of a dashboard allows you to save, preview, and edit the dashboard, as shown in Dashboard toolbar.

Dashboard toolbar

7.4.3.4.3. Adjust chart positions

Multiple charts may be displayed on the same dashboard. In this case, you can drag the charts to adjust their positions.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Select a chart or widget.
3. Drag the chart or widget to the desired position.

> **Note** You can drag a chart or widget to anywhere within the display area of the dashboard.

7.4.3.4.4. View chart data

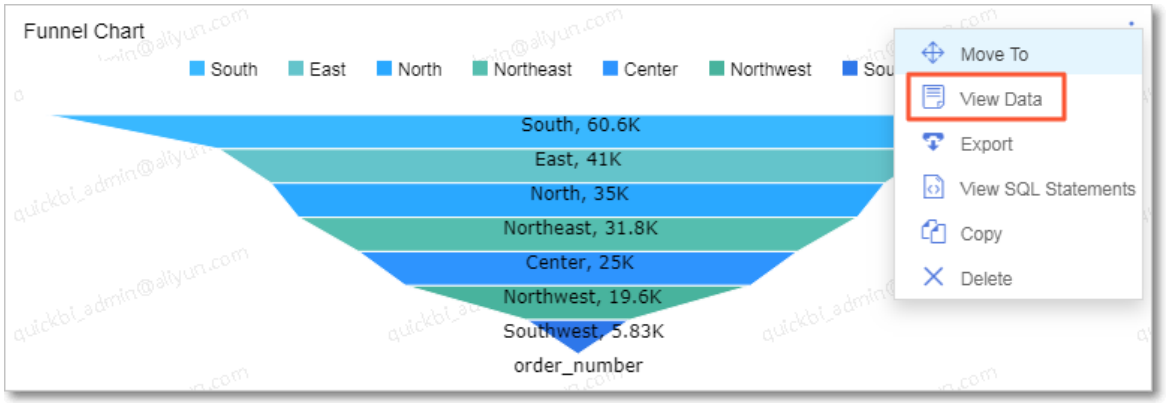
This topic describes how to view chart data.

Prerequisites

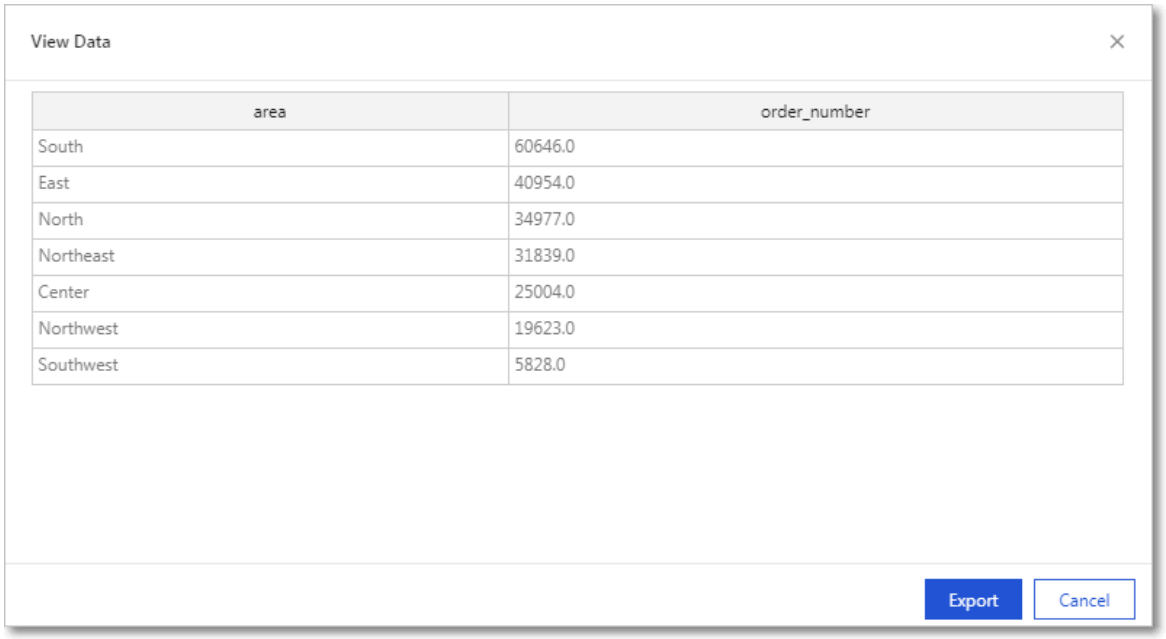
The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Select a chart, for example, a funnel chart.
3. Click the More icon in the upper-right corner of the chart.
4. Select **View Data** to view data items in the chart.



5. Select **Export** to download data to a local PC.



7.4.3.4.5. Change chart types

You can select different chart types from the toolbar on the top of the dashboard. This topic describes how to change chart types.

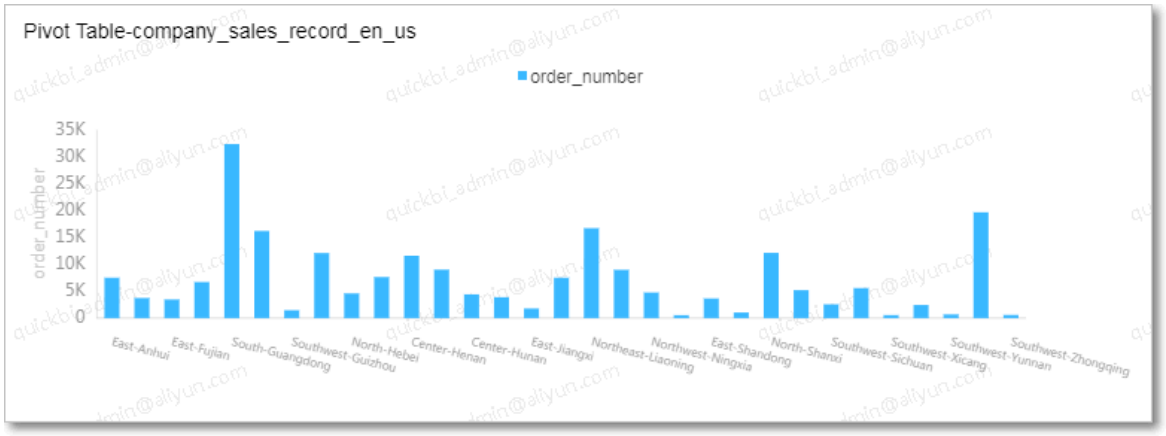
Prerequisites

The Quick BI service is purchased.

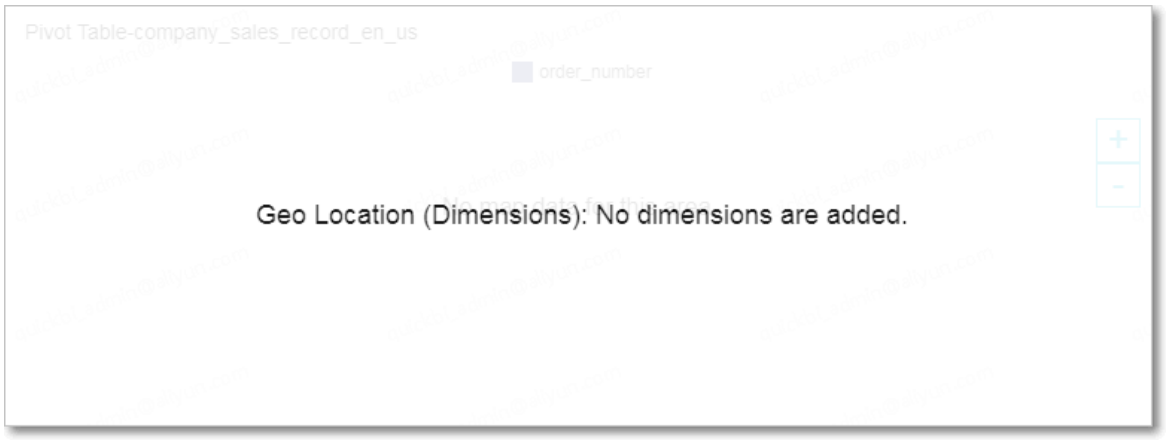
Procedure

1. Go to the target dashboard.
2. Select a chart, such as a cross table.
3. In the Graphic Design area, click **Change Chart Type** and select the required type, such as a vertical bar chart.
4. Click the **Vertical Bar Chart** icon to change the chart type.

   The system then converts the cross table to a vertical bar chart.



If the switch between chart types fails, the elements of the selected chart type do not match those of the current chart type. You need to modify the problematic data fields based on the requirements of the selected chart type.



You can follow the instructions to adjust the dimensions and measures to change the chart type.

7.4.3.4.6. Add to Favorites

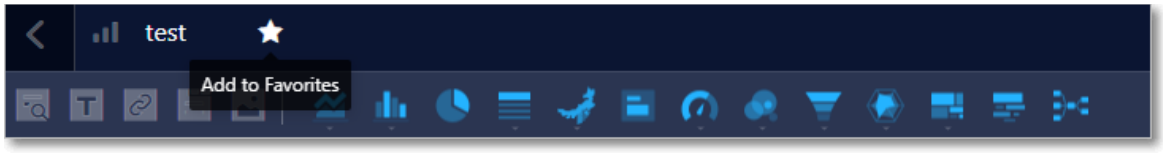
This topic describes how to add a dashboard to the Favorites tab.

Prerequisites

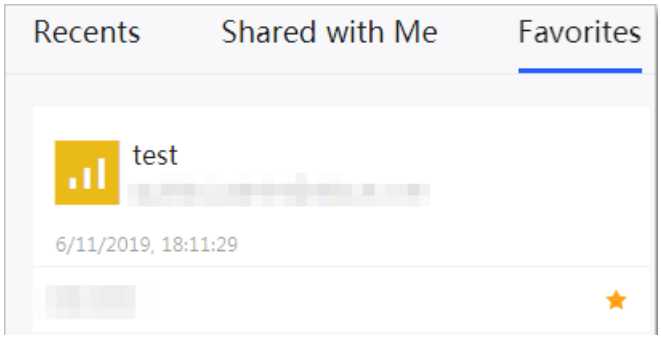
The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. On the top of the dashboard, click the **Add to Favorites** icon.



3. On the Quick BI homepage, you can click the **Favorites** tab to view the dashboards that you have added.



7.4.3.4.7. Delete a chart

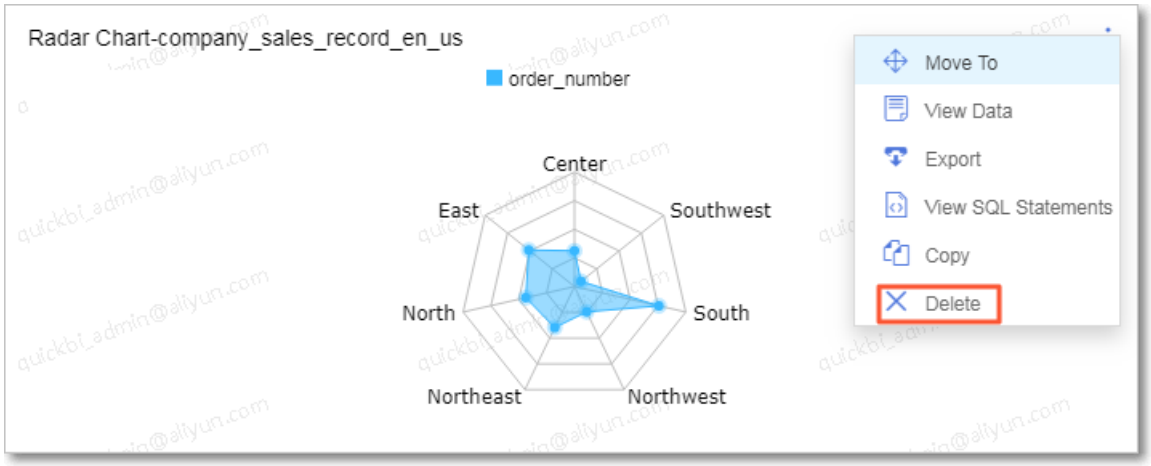
This topic describes how to delete a chart.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Select a chart, such as a radar chart.
3. Click the **More** icon in the upper-right corner of the chart.
4. Select **Delete**.



7.4.3.4.8. Widgets

7.4.3.4.8.1. Overview

This topic describes the widgets of a dashboard.

The display area of a dashboard provides the following widgets:

- Filter Bar
- Text Area
- IFrame
- Tab
- Image

7.4.3.4.8.2. Filter bar

In a dashboard, you can use the Filter Bar widget to query data in one or more charts.

7.4.3.4.8.3. Expanded filter bar

7.4.3.4.8.4. Compound Query Control

7.4.3.4.8.5. Text Area

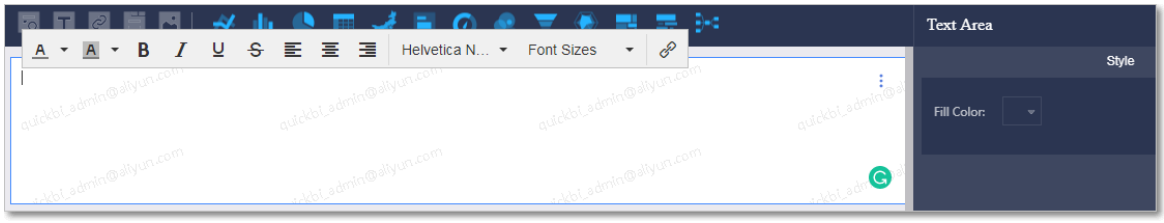
The Text Area widget allows you to enter text into a text area, for example, a title for a chart.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Click the **Text Area** icon.
3. Enter text in the text box.



7.4.3.4.8.6. IFrame

You can use the IFrame widget to insert web pages to query Internet data or browse web pages or websites related to the data on the current dashboard in real time.

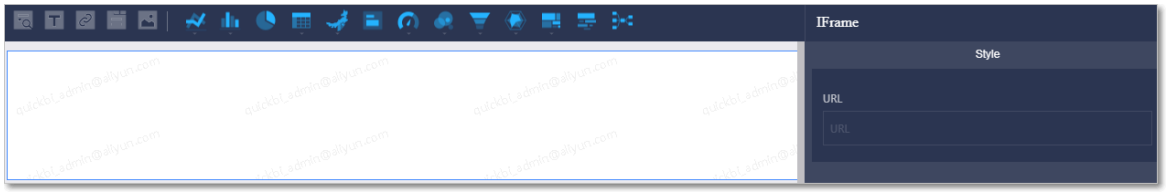
Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Click the **IFrame** icon.

3. In the URL input box, enter the address of the web page you want to visit.



> **Note** The web page address must start with https.

If you want to delete the current IFrame widget, click the **More** icon in the upper-right corner of the **IFrame** widget and select **Delete**.

7.4.3.4.8.7. Tab

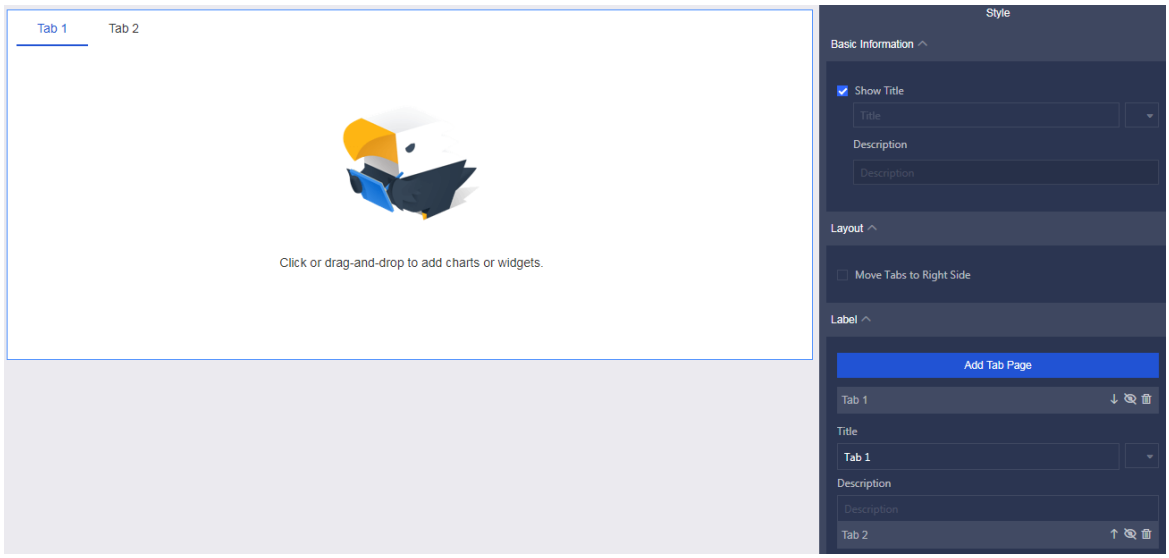
The tab widget enables you to display charts as tabs on a dashboard.

Prerequisites

The Quick BI service is purchased.

Procedure

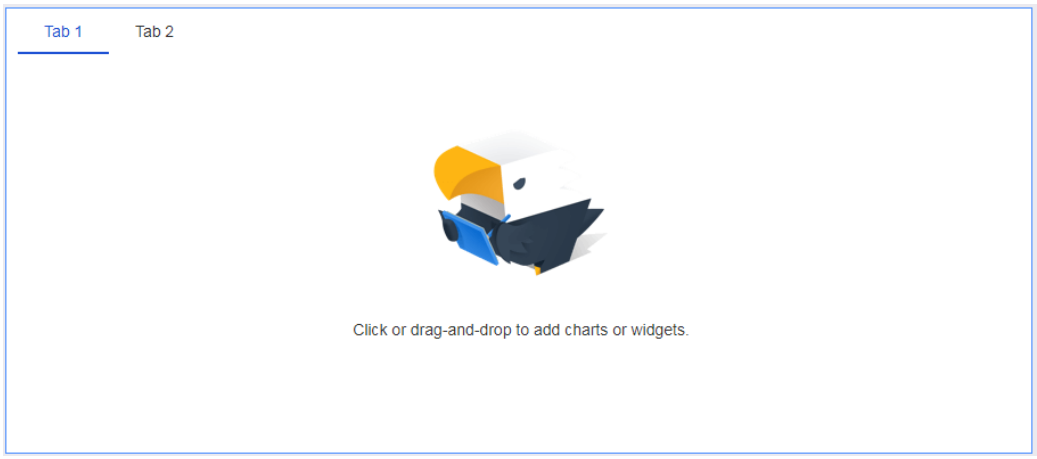
1. Go to the target dashboard.
2. Click the Tab icon.
3. Click the **Style** tab. Click **Add Tab Page** in the **Label** field.



4. Select a tab where you want to add charts, as shown in Tabs.

   Click **Tab 1**. The tab name **Tab 1** becomes blue.

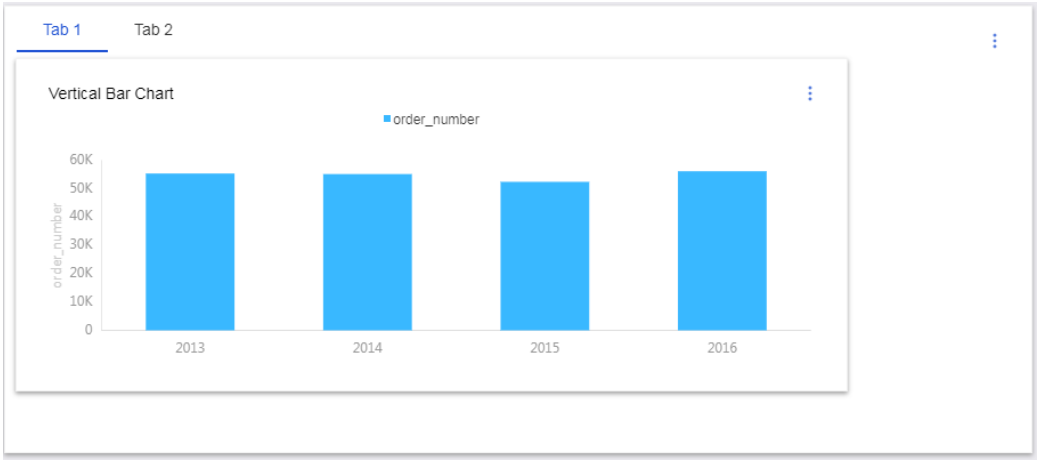
   Tabs



5. Click a chart icon to add a chart to Tab 1.

Add and configure a chart. Sample tab shows the tabs.

Sample tab



If you want to delete the current tab widget, click the **More** icon in the upper-right corner of the tab, and click **Delete**.

7.4.3.4.8.8. Image

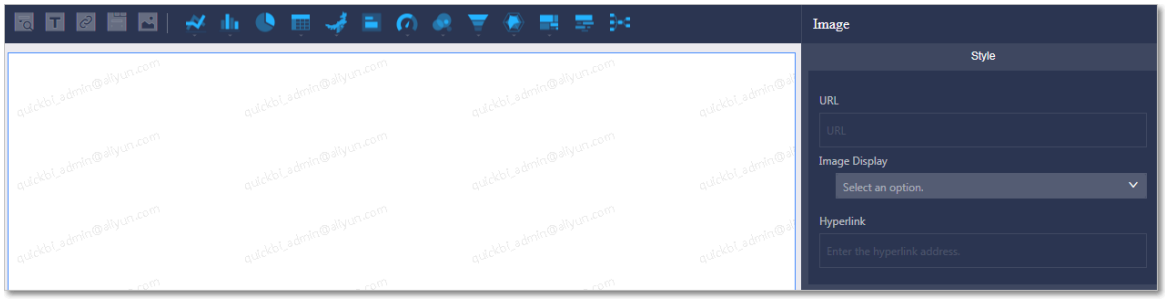
The Image widget allows you to insert images into a dashboard. You can adjust the image position and display effects as needed.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Go to the target dashboard.
2. Click the **Image** icon.
3. In the Style field, enter the URL and hyperlink of the image, and specify **Image Display**.



7.4.4. Create a chart on the dashboard

7.4.4.1. Create a line chart

A line chart shows data change trends and the interactions between multiple groups of data over a period of time. For example, you can use a line chart to analyze the sales volumes of a group of products or multiple groups of products over a period of time to forecast future sales volumes.

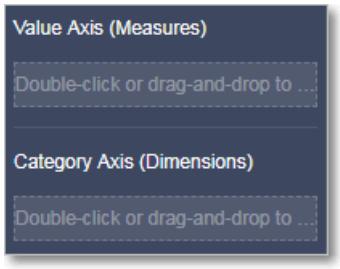
Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A line chart consists of a category axis and a value axis. The category axis is horizontal and determined by dimensions, such as date, province, and product type. The value axis is vertical and determined by measures, such as order quantity and performance metrics.

The system automatically matches dimensions with the category axis and measures with the value axis. You only need to follow the instructions to add fields.



You must specify at least one dimension for the category axis and at least one measure for the value axis. You can specify only one dimension for the color legend.

> Note The color legend is applicable only when the value axis has one measure.

The following example uses the company_sales_record dataset to describe how to use a line chart to demonstrate the order quantity of each type of products in different provinces every year.

Procedure

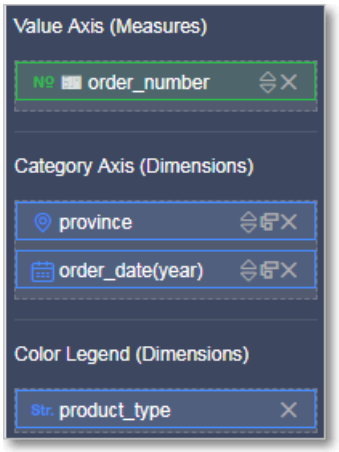
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Line Chart** icon.
5. Click the **Data** tab, and select required dimensions and measures.

   In the Dimensions list, find and add **order_date(year)** and **province** to the Category Axis (Dim.) field. In the Measures list, find and add **order_number** to the Value Axis (Mea.) field, as shown in Specify fields for the line chart.

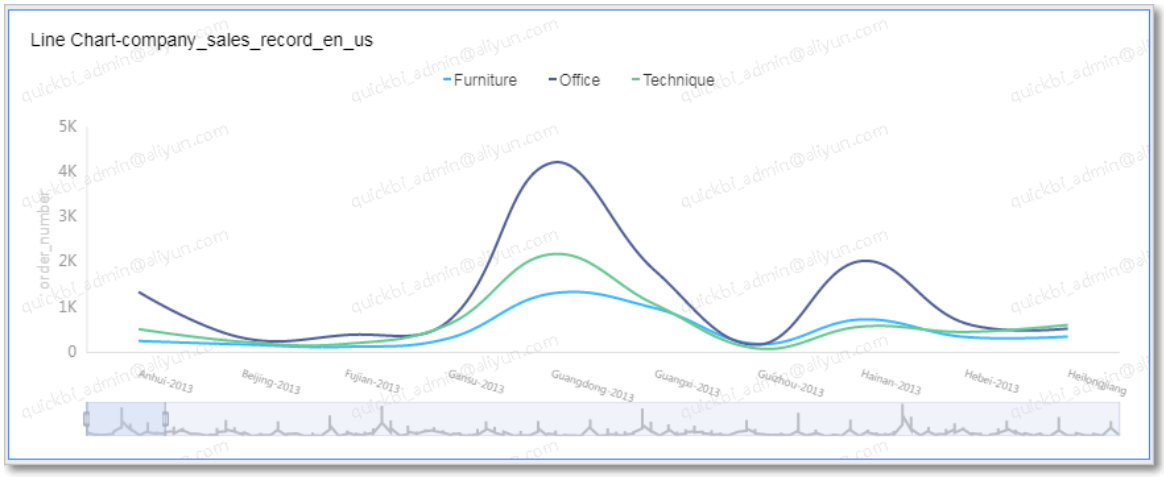
   > Note Ensure that you have converted the dimension type of province from String to Geo.

   For information about how to convert a dimension type, see Edit a dimension.

   Specify fields for the line chart

6. Drag **product_type** to the **Color Legend (Dim.)** field.
7. Click **Update**. The chart is updated.
8. Click the **Style** tab, and change the title, layout, and legend position of the chart.



9. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click **OK**.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.2. Create an area chart

An area chart shows data change trends at equal intervals with different sizes of area. In addition, you can use an area chart to analyze the interactions between multiple groups of data over a period of time. For example, analyze the sales volumes of a group of products or multiple groups of products over a period of time to forecast future sales volumes.

An area chart consists of a category axis and a value axis. The category axis is horizontal and determined by dimensions, such as date, province, and product type. The value axis is vertical and determined by measures, such as order quantity and performance metrics.

The system automatically matches dimensions with the category axis and measures with the value axis. You only need to follow the instructions to add fields.

Notes

You must specify at least one dimension for the category axis and at least one measure for the value axis.

> **Note** The color legend is applicable only when the value axis has one measure.

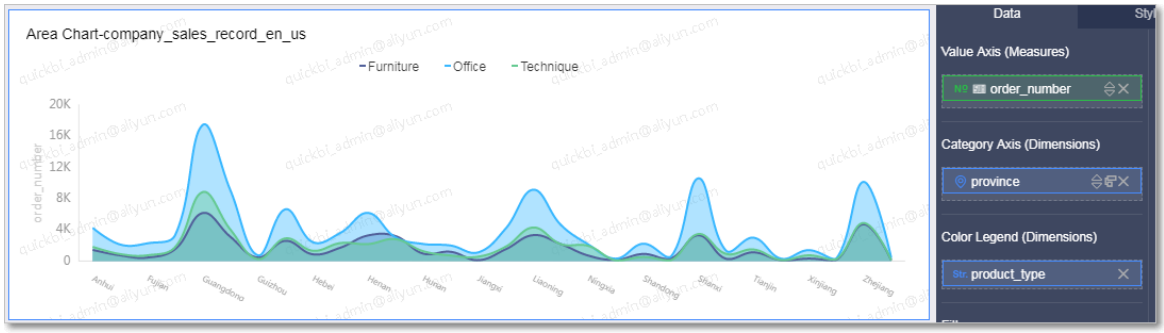
The following example uses the company_sales_record dataset to describe how to use an area chart to demonstrate the order quantity of each type of products in different provinces.

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Area Chart** icon. An area chart appears in the display area of the dashboard.
5. Select required dimensions and measures.

In the Dimensions list, find and add **province** to the Category Axis (Dim.) field. In the Measures list, find and add **order_number** to the Value Axis (Mea.) field.

> **Note** Ensure that you have converted the dimension type of province from String to Geo.

6. Drag **product_type** to the Color Legend (Dim.) field. Click **Update**.
7. Click the **Style** tab, and change the title, layout, legend position, and axis style of the chart. The following figure shows an updated area chart.



> **Note** You can switch to another area chart type, such as stacked area chart, 100% stacked area chart, or stacked line chart as required.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click **OK**.

If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.3. Create a vertical bar chart

A vertical bar chart demonstrates data changes over a period of time or comparisons between objects. For example, you can use a vertical bar chart to show the traffic flow in different time periods of time at a crossing.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

Similar to a line chart, a vertical bar chart consists of a category axis and a value axis.

This topic uses the following scenarios to describe how to use the filter and the Dual Y-Axis function in a vertical bar chart.

- Scenario 1: Compare the shipping costs for different products in provinces in the East China region.
- Scenario 2: Compare the order quantities and average profits of different products in different provinces.

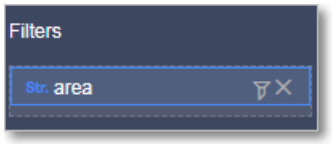
You must specify at least one dimension, such as **province** or **product_type**, for the category axis. You must specify at least one measure, such as **order_number** or **profit_amt**, for the value axis. You can specify only one dimension for the color legend.

> **Note** The color legend is applicable only when the value axis has one measure.

Procedure

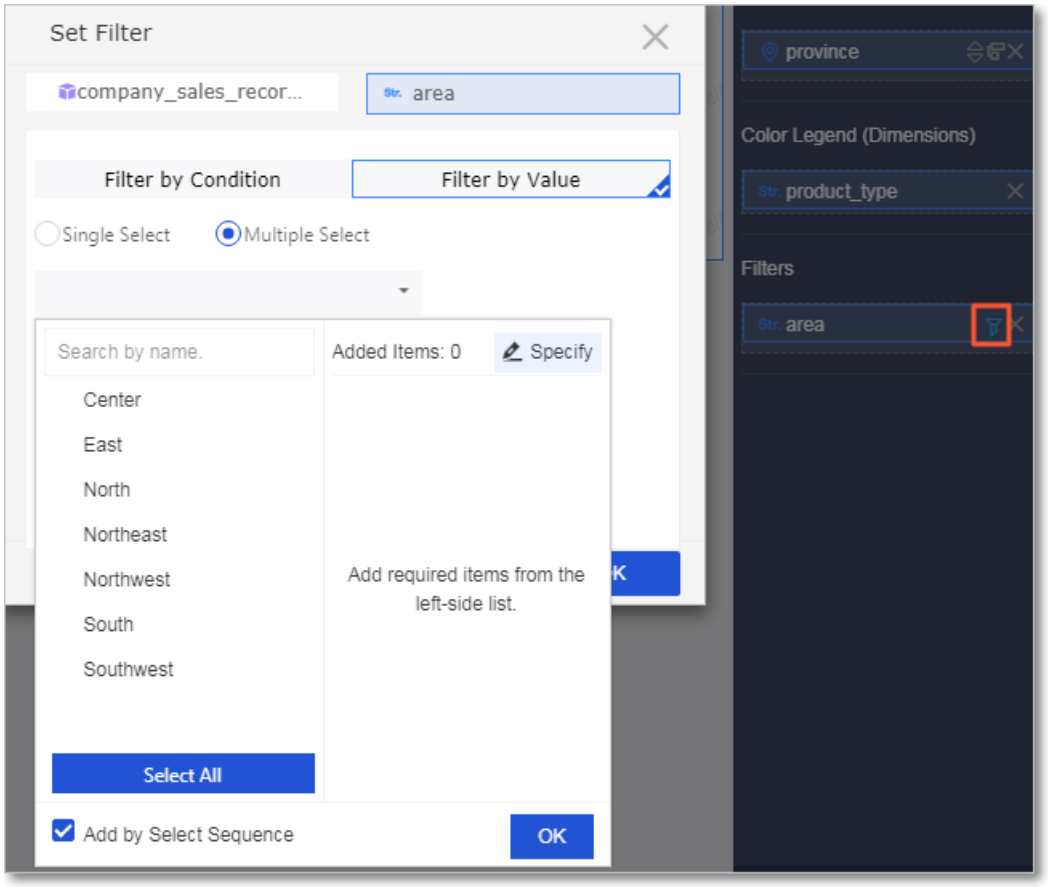
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Vertical Bar Chart** icon. Scenario 1: The following example uses the company_sales_record dataset to describe how to use a vertical bar chart to compare the shipping costs for different products in provinces in the East China region.
   i. In the Dimensions list, find and add area to the Filters field, as shown in Filters. You can use the filter to filter data of the **East China** region.

      Filters

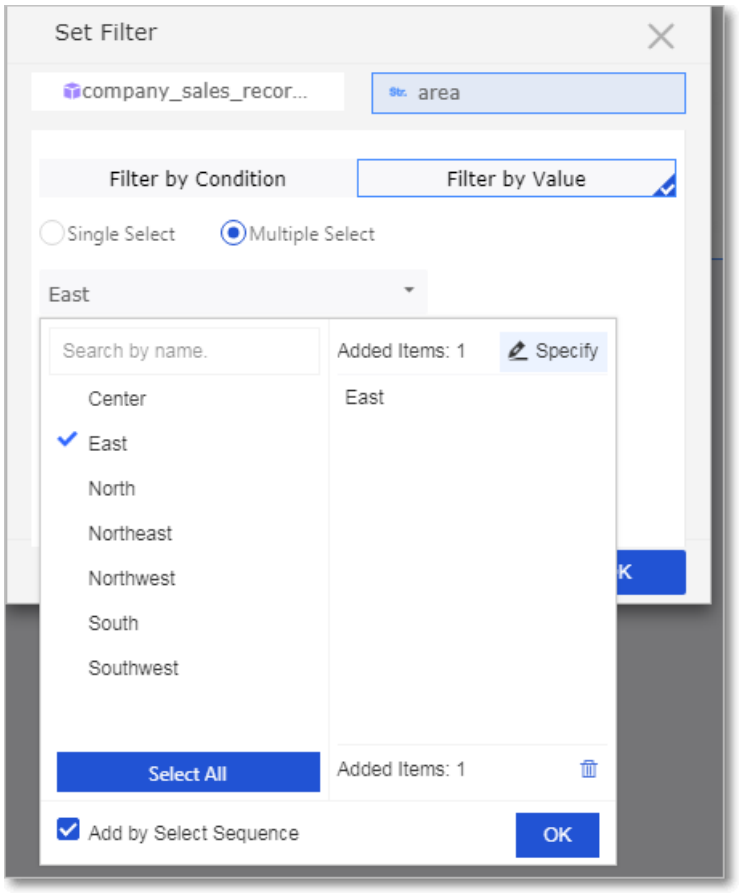
      

   ii. Click the **Filter** icon.

iii. In the Set Filter dialog box that appears, select Filter by Value and **Multiple Select**. The system automatically lists all available options.



iv. Select **East** and click **OK**.



v. In the Dimensions list, find and add **province** and **product_type** to the Category Axis (Dim.) field.

vi. In the Measures list, find and add **shipping_cost** to the Value Axis (Measures) field.

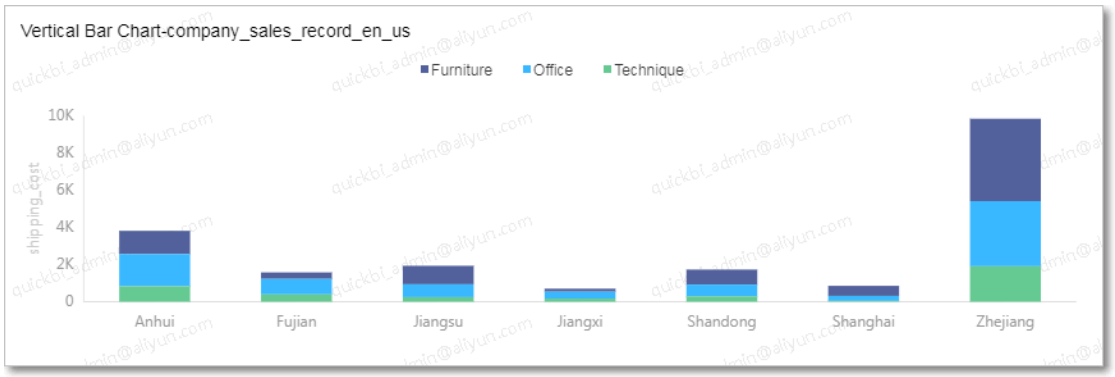
> **Note** Ensure that you have converted the dimension type of province from String to Geo.

For information about how to convert a dimension type, see Edit a dimension.

vii. Drag **product_type** from the Category Axis (Dim.) field to the **Color Legend (Dim.)** field.

viii. Click **Update**. The chart is updated.

ix. Click the **Style** tab and select **Stacked** in the Chart Type section.

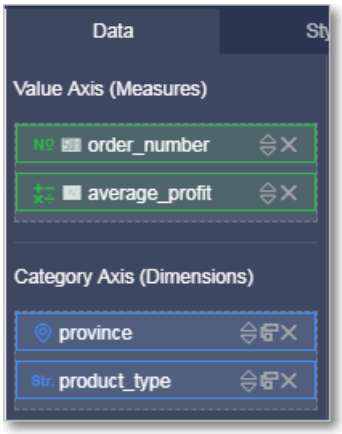


Scenario 2: The following example uses the company_sales_record dataset to describe how to use a vertical bar chart to compare the order quantities and average profits of different products in different provinces.

> **Note**
>
> Data modeling may be required in this scenario. For information about data modeling, see Add a calculated field.

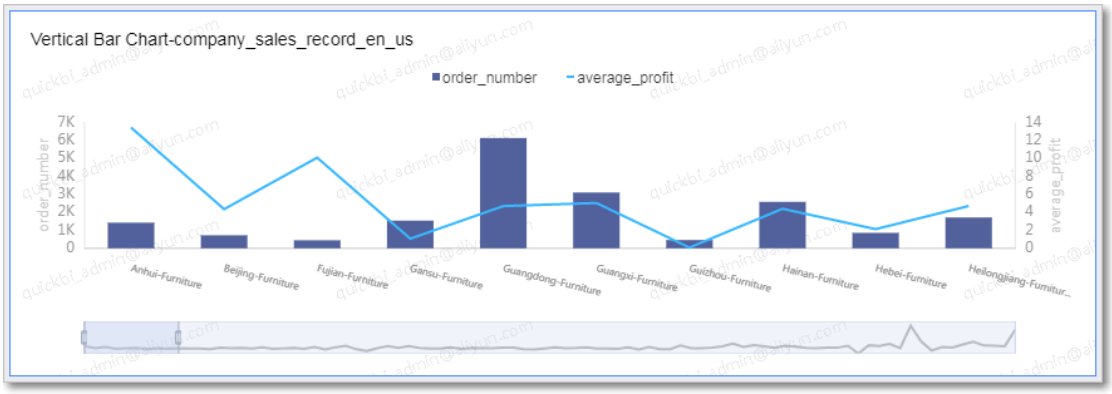
i. Click the **Data** tab and select required dimensions and measures.

In the Dimensions list, find and add **province** and **product_type** to the Category Axis (Dim.) field. In the Measures list, find and add **order_number** and **average_profit** to the Value Axis (Mea.) field.



ii. Click **Update**. The chart is updated.

iii. Click the **Style** tab and select **Dual Y-Axis** in the Chart Type section.



iv. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click OK.

7.4.4.4. Create a waterfall chart

A waterfall chart reflects data changes at different time periods or under the impact of different factors. It uses a combination of absolute and relative values and is suitable for business analysis and financial analysis.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

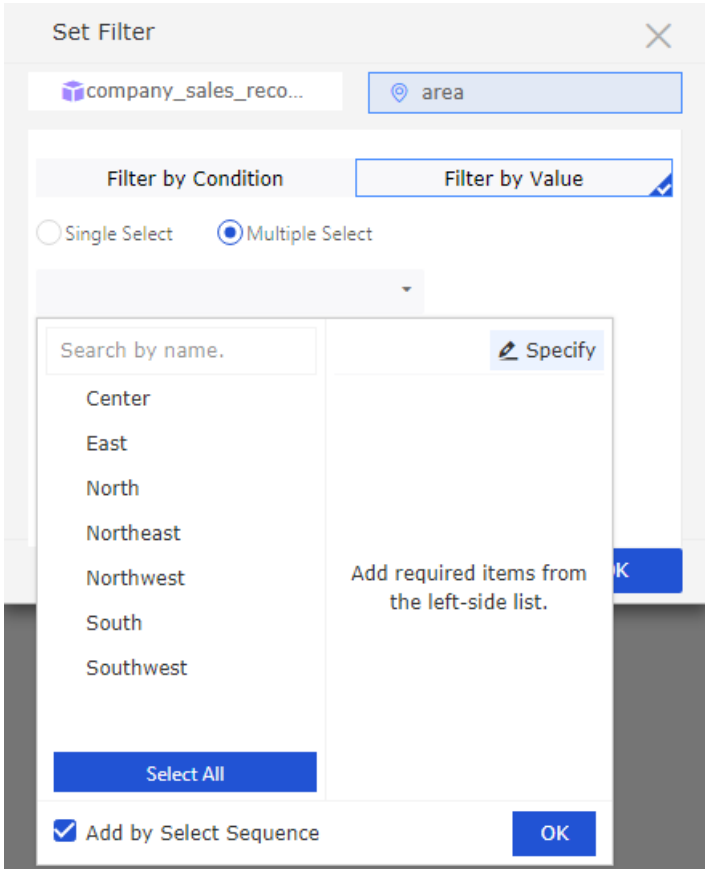
Context

You must specify at least one dimension, such as province or product type, for the category axis. You can specify only one measure, such as order quantity or profit, for the value axis.

The following example uses the *company_sales_record dataset* to describe how to use a waterfall chart to compare the shipping costs in the provinces of East China.

Create a waterfall chart

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the **My Items** tab of the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Waterfall Chart** icon.
5. Click the **Data** tab and select the required measure. In the Dimensions list, find and add **province** to the Category Axis (Dim.) field and **area** to the Filters field. In the Measures list, find and add **shipping_cost** to the Value Axis (Mea.) field.
6. Click . In the **Set Filter** dialog box that appears, click the Filter by Value tab and select **North**.



7. Click **Update**. The chart is updated.
8. Click the **Style** tab and set parameters in the **Basic Information**, **Parameter Settings**, **Chart Type**, **Axes**, and **Series Settings** sections.
9. Click Save in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click **OK**.

   If you want to delete the chart, click the **More** icon ( ) in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

You can perform the following steps on the **Style** tab:

1. In the **Basic Information** section, set Show Title and Description, Show Link, and Background. In this example, **Dark Color** is selected for **Background**.

   > **Note** If you want to redirect to a report or an external page, select **Show Link** and specify Link Text and Link Address.

2. In the **Parameter Settings** section, configure Show Initial Value, Show Cumulative Value, and Show Initial Value/Cumulative Value Labels. In this example, Show Initial Value/Cumulative Value Labels is selected.
3. In the **Chart Type** section, configure Show Labels, Show Legend, and Color Settings. In this example, **Show Labels** is selected and the color legend is displayed on the top of the chart.

   > **Note** If multiple measures are specified for the chart, all measure labels are displayed after you select Show Labels. Labels can be displayed in two modes: Smart Display and Full Display. Assume that a chart has many dimension values and the scroll bar is not shown in the chart. In Smart Display mode, only part of labels are displayed. In Full Display mode, all labels are displayed.

4. In the **Axes** section, set Axis Title and Unit. In this example, **Show Scale** is selected on the X-Axis tab.
5. In the **Series Settings** section, set Alias and Data Display Format for a measure. In this example, the default data display format **AutoFit** is used.

After you complete the configurations, the chart is shown in the following figure.

Configure filter interaction

On the **Advanced** tab, you can set Auto Refresh and Filter Interaction. The following example uses a pie chart to describe how to configure filter interaction.

1. Click the **Pie Chart** icon. A pie chart appears in the display area of the dashboard.
2. Specify measures and dimensions.

   > **Note** The measures and dimensions of the pie chart can be different from the waterfall chart.

3. Select the waterfall chart and click the **Advanced** tab.
4. On the **Advanced** tab, click the Edit icon next to Filter Interaction. In the Filter Interaction dialog box that appears, select fields and charts for interaction.
5. Click **OK**.

The following figure shows the chart that appears after the configuration is complete. After you click an area in the waterfall chart, the pie chart displays the data corresponding to the area.

> **Note** The initial value and cumulative value do not support filter interaction.

7.4.4.5. Create a horizontal bar chart

Similar to a vertical bar chart, a horizontal bar chart displays the differences between data of different categories.

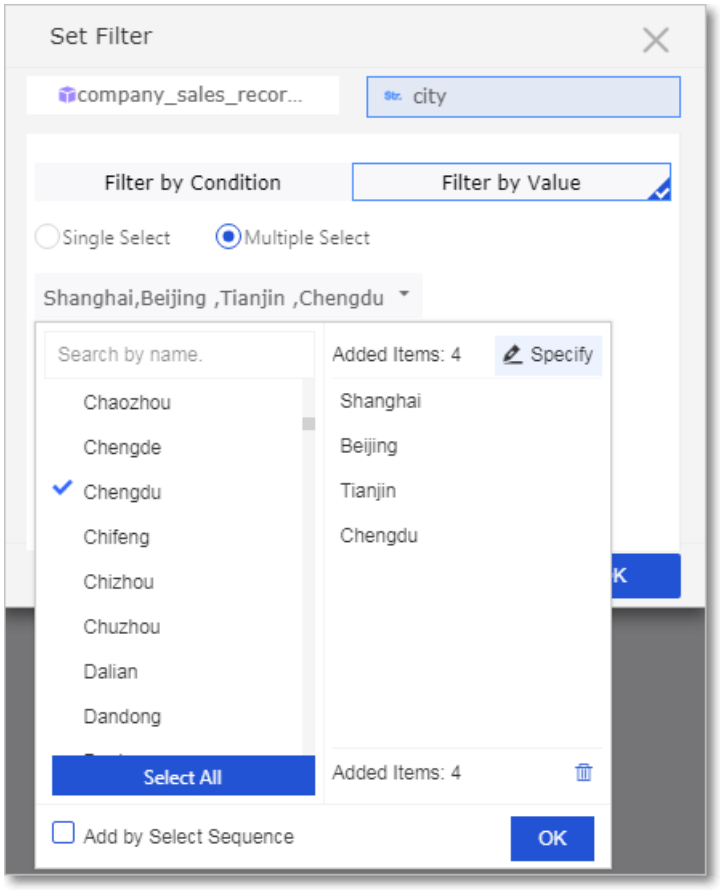
Notes

You must specify at least one dimension, such as province or product type, for the category axis and specify at least one measure, such as order quantity or profit, for the value axis. You can specify only one dimension for the color legend.

> **Note** The color legend is applicable only when the value axis has one measure.

The following example uses the company_sales_record dataset to describe how to use a horizontal bar chart to compare the shipping costs of different products in different municipalities.

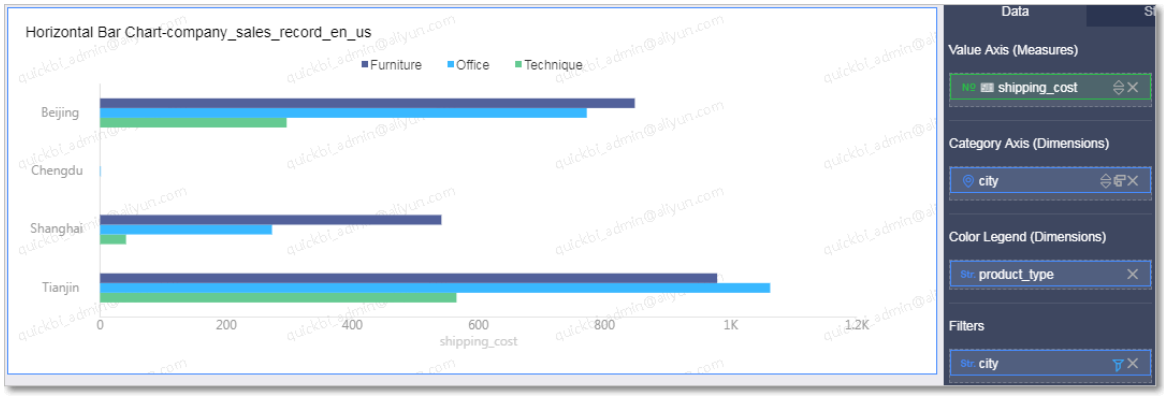
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Horizontal Bar Chart** icon. A horizontal bar chart appears in the display area of the dashboard.
5. In the Dimensions list, find and add **city** to the Filters field and select four municipalities, as shown in the following figure.



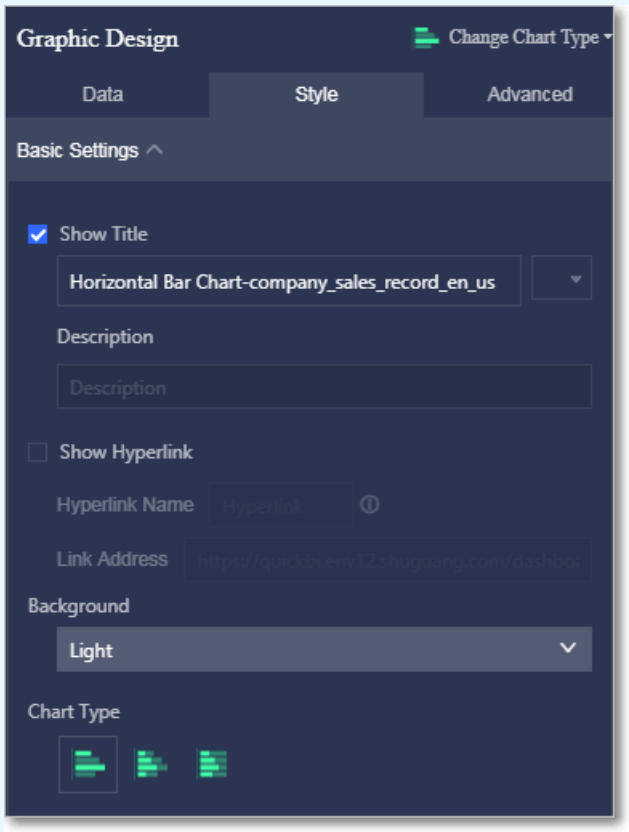
6. In the Dimensions list, find and add **city** to the Category Axis (Dim.) field. In the Measures list, find and add **shipping_cost** to the Value Axis (Mea.) field. In the Dimensions list, find and add **product_type** to the Color Legend (Dim.) field.

> **Note**
>
> - Ensure that you have converted the dimension type of city from String to Geo.
> - The color legend is applicable only when the value axis has one measure.

7. Click **Update**. The chart is updated.



Note You can also switch the current chart to another bar chart type, such as a stacked horizontal bar chart or 100% stacked horizontal bar chart, as shown in the following figure.



8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click **OK**.

If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.6. Create a progress bar

Similar to a gauge, a progress bar displays the progress of a specific metric.

A progress bar has a pointer. The pointer is determined by measures, such as order quantity.

Notes

- You can specify one to five measures for the pointer.
- To use a progress bar, you must set the maximum and minimum values in the **Series Settings** section on the **Style** tab.

The following example uses the company_sales_record dataset to describe how to use a progress bar to show order completion.

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the

Create Dashboard icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.

4. On the dashboard edit page, click the Progress Bar icon. A progress bar appears in the display area of the dashboard.
5. Click the **Data** tab and select required measures.

   In the Measures list, find and add **order_number** to the Pointer (Mea.) field. Click the **Style** tab, configure the chart title and color legend position, set an alias for the measure, and set the maximum and minimum values.
6. Click **Update**. The chart is updated.

7. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard and click **OK**.

If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.7. Create a combination chart

A combination chart compares data across multiple categories and magnitudes.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

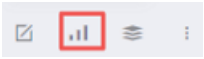
Context

A combination chart provides a dual y-axis to display data in multiple chart types, such as line chart, vertical bar chart, and area chart, in stacked and 100% stacked modes. For example, you can use a combination chart to display the change trends of different projects.

A combination chart consists of a category axis, primary value axis, and secondary value axis.

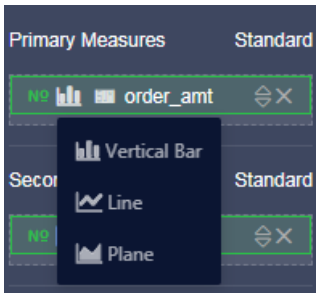
> **Notice** You can specify at least one dimension for the category axis, such as the order_date(year). You can specify at least one measure each for the primary and secondary value axes, such as order price and profit. You can specify only one dimension for the color legend. The legend is applicable only when the primary or secondary value axis has one measure.

Procedure

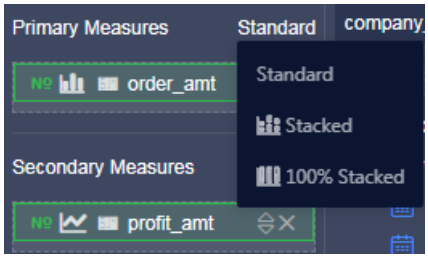
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the *company_sales_record* dataset, click  in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click  .
5. Click the **Data** tab and select required dimensions and measures.

   In the Dimensions list, find and add **order_date (year)** to the Category Axis (Dim.) field. In the Measures list, find and add **order_amt** to the Primary Measures field and **profit_amt** to the Secondary Measures field.

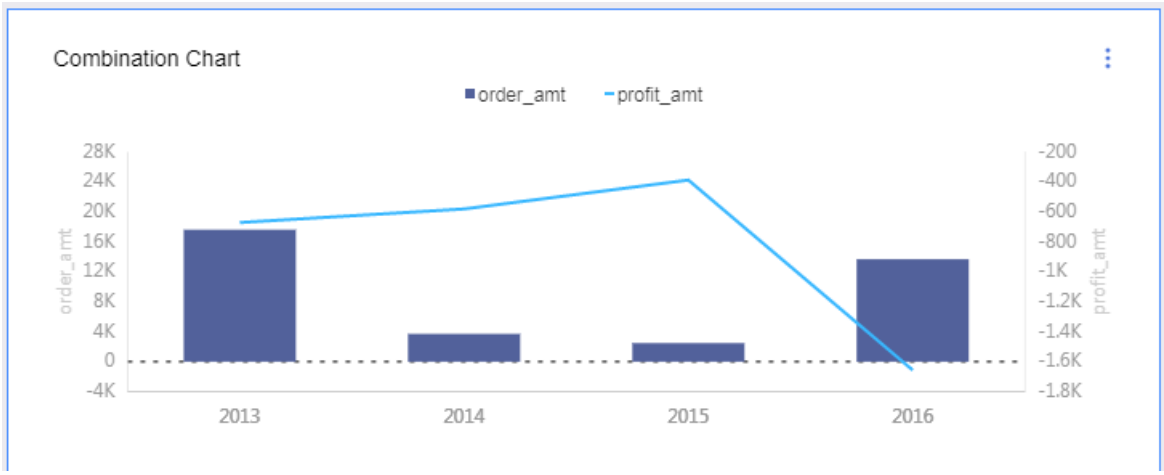
   - In the Primary Measures and Secondary Measures fields, you can click  to switch the chart type to vertical bar chart, line chart, or area chart.

   - You can click  to switch the stack mode.

6. Click **Update**. The chart is updated.

7. Click **Save**. The **Save Dashboard** dialog box appears. Set Name and Save To.
8. Click **OK** to save the dashboard.

   If you want to delete the chart, follow these steps:

   i. Click ⋮ in the upper-right corner of the chart.

   ii. Select **Delete**.

7.4.4.8. Create a pie chart

A pie chart shows data of different objects. Each object has a unique color or pattern. A pie chart shows the ratios of multiple values to the total amount. For example, you can use a pie chart to show the ratio of the income tax to the total personal income or the ratio of the sales volume of a car brand to the total sales volume.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A pie chart consists of multiple slices. Slice labels are determined by a dimension, such as area or product type. The central angle of each slice is determined by a measure, such as order quantity, order price, or profit.

You can specify only one dimension for slice labels and only one measure for central angles.

The following example uses the company_sales_record dataset to describe how to use a pie chart to compare the shipping costs in different regions.

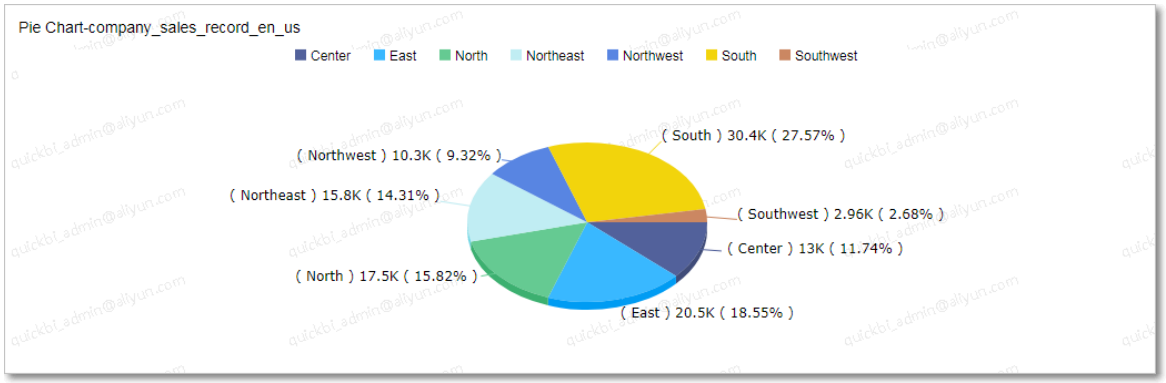
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Pie Chart** icon.
5. Click the **Data** tab and select required dimension and measure.

   In the Dimensions list, find and add area to the Labels (Dim.) field. In the Measures list, find and add shipping_cost to the Central Angle (Measures) field, as shown in Specify fields for the pie chart.

   Specify fields for the pie chart

6. Click **Update**. The chart is updated.
7. Click the **Style** tab. Select **3D** for **Display Mode** and **Name, Percentage** for **Label Style**.



8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.9. Create a bubble map

A bubble map uses a map profile as its background and shows data distribution with bubbles of different sizes. It displays data metrics and their distribution in a country or region. For example, you can use a bubble map to display the volume of tourist arrivals at different destinations, or the average income in different regions.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A bubble map consists of geographic locations and the bubble size. Geographic locations are determined by a dimension, such as province. The bubble size is determined by measures, such as shipping cost and order quantity.

You can specify only one dimension, such as area, province, or city, for geographic locations. The dimension type must be Geo. You can specify one to five measures for the bubble size.

The following example uses the company_sales_record dataset to describe how to use a bubble map to compare the order quantities and average profits in different provinces.

Data modeling may be required for this scenario.

For information about data modeling, see Add a calculated field.

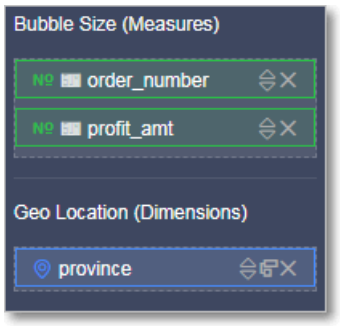
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. In the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Bubble Map** icon.
5. Click the **Data** tab and select required dimension and measures.

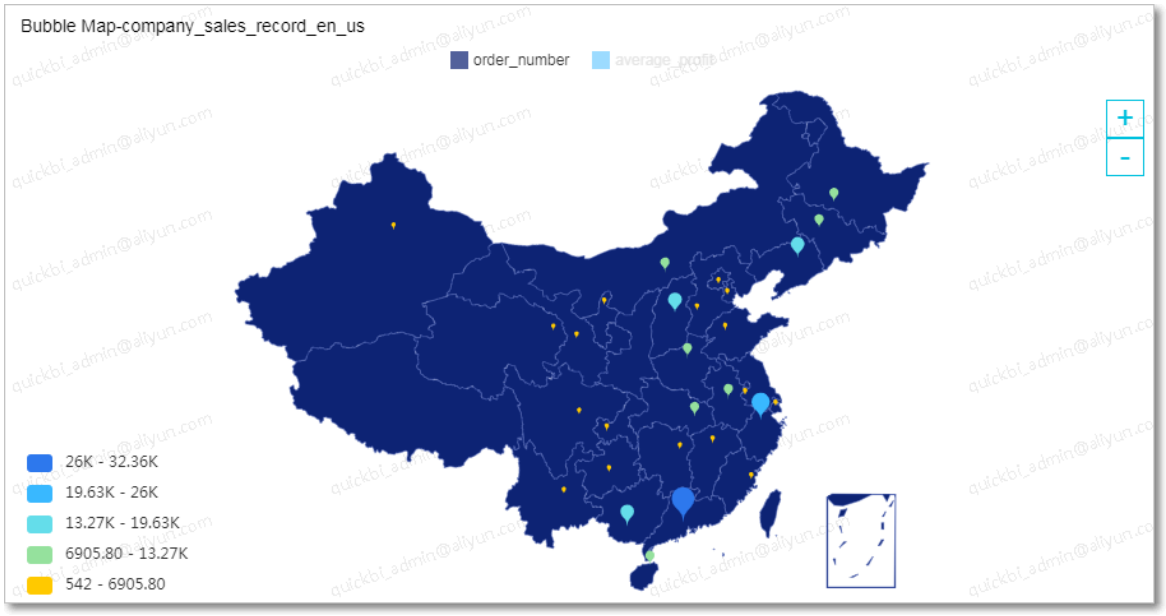
   In the Dimensions list, find and add province to the Geo Location (Dim.) field. In the Measures list, find and add order_number and average_profit to the Bubble Size (Mea.) field, as shown in Specify fields for the bubble map.

   > **Note** Ensure that you have converted the dimension type of province from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

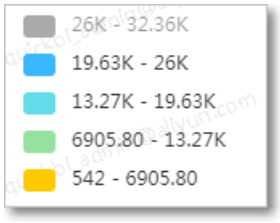
   Specify fields for the bubble map

6. Click **Update**. The chart is updated.
7. Click the **Style** tab, and edit the title and legend position of the chart.



- You can switch between order_number and average_profit to show required data.
- Click a legend to hide its data.



- Click the plus (+) or minus (-) sign to zoom in or zoom out on the map.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.10. Create a colored map

Similar to a bubble map, a colored map displays data in a single color at different saturation levels. This visualization is useful to show the distribution of data across areas on a map.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A colored map consists of geographic locations and a colorscale. Geographic locations are determined by a dimension, such as province. The colorscale is determined by measures, such as order price and profit.

You can specify only one dimension for geographic locations. The type of the dimension must be Geo. You can specify one to five measures for the colorscale.

The following example uses the company_sales_record dataset to describe how to use a colored map to compare the shipping costs, order price, and profits in different regions.

Procedure

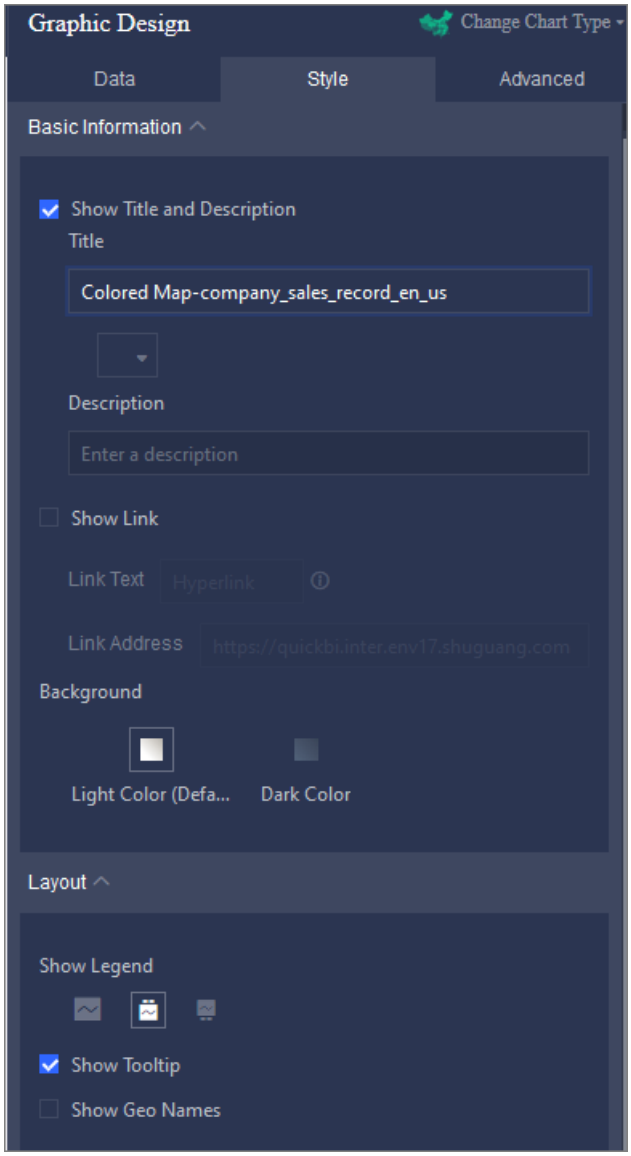
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Colored Map** icon.
5. Click the **Data** tab and select required dimension and measures.

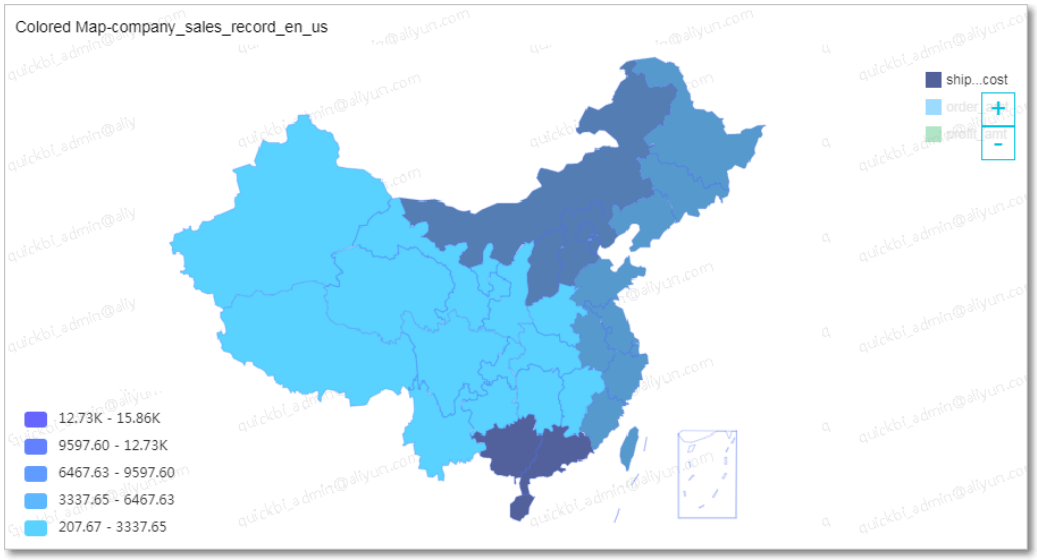
   In the Dimensions list, find and add area to the Geo Location (Dim.) field. In the Measures list, find and add order_amt, profit_amt, and shipping_cost to the Colorscale (Mea.) field.

> **Note** Ensure that you have converted the dimension type of area from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

6. Click **Update**. The chart is updated.
7. Click the **Style** tab, click the **Right** icon for the **Show legend** field in the **Layout** section, as shown in The colored map.

The colored map





You can perform various operations on the map, such as change the title, adjust the position and size, and hide irrelevant data. For more information, see Create a bubble chart.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.11. Create a geo bubble map

A geo bubble map uses a map profile as its background and shows data distribution with bubbles of different sizes. It displays data metrics and distribution in a country, region, or city. Compared with bubble maps, geo bubble maps provide more accurate geographic locations.

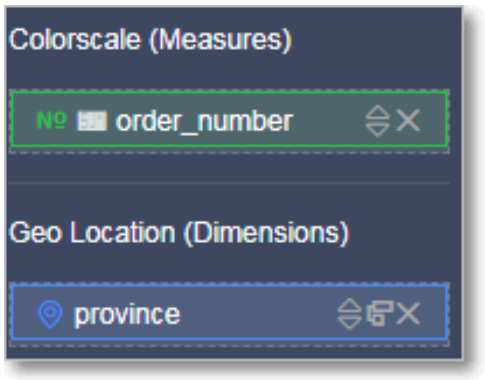
A geo bubble map consists of geographic locations displayed with different saturation levels. Geographic locations are determined by a dimension, such as province. Saturation levels are determined by a measure, such as order quantity. You must specify one dimension and one measure for a geo bubble map. The dimension type must be Geo.

The following example uses the company_sales_record dataset to describes how to use a geo bubble map to compare the order quantities of different provinces in the North China region.

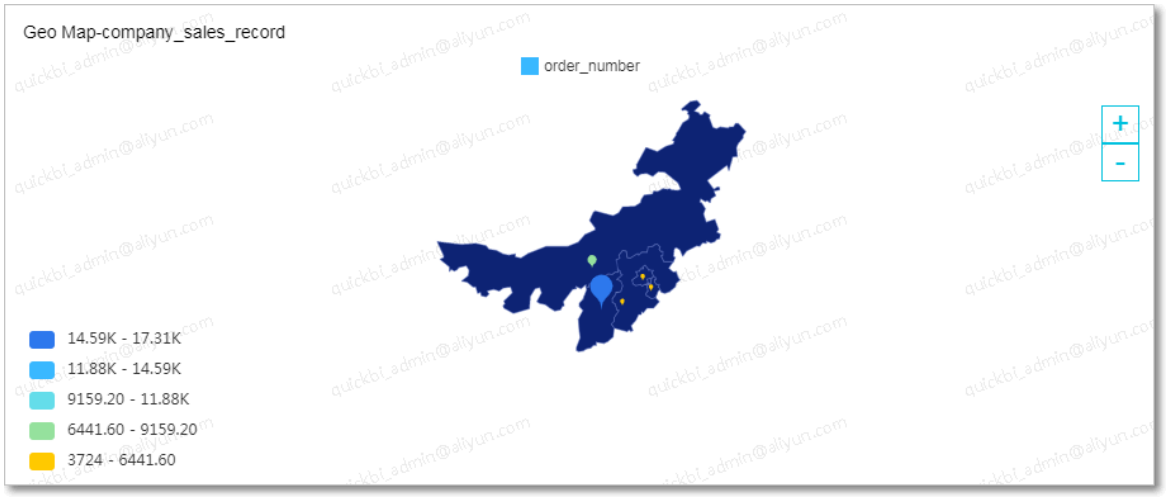
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets** to go to the Datasets page.
3. Find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column, select Standard in the dialog box that appears, and click OK.
4. Click the Geo Bubble Map icon. A geo bubble map is created in the display area of the dashboard.
5. On the **Data** tab, select the target measure and dimension.

   In the Dimensions list, find and add the **province** dimension to the Location (Dimension) field. In the Measures list, find and add the **order_number** measure to the Color Scale (Measures) field, as shown in the following figure.

> **Note** Ensure that you have converted the dimension type of province from String to Geo.



6. Click **Update**. The chart is updated.
7. On the **Style** tab, change the value ranges and saturation levels, as shown in the following figure.



8. Click **Save** to save the dashboard.

If you want to delete the geo bubble map, click the **More** icon in the upper-right corner of the map and select **Delete**.

7.4.4.12. Create a geo map

A geo map displays data as a single color at different saturation levels. This visualization is useful to show distribution of data across areas on a map. Compared with colored maps, geo maps provide more accurate geographic locations.

A geo map consists of geographic locations displayed in one color with different saturation levels. Geographic locations are determined by a dimension, such as province. Saturation levels are determined by a measure, such as order quantity. You must specify one dimension and one measure for a geo map. The dimension type must be Geo.

The following example uses the company_sales_record dataset to describe how to use a geo map to compare the order quantities of different provinces in the South China region.

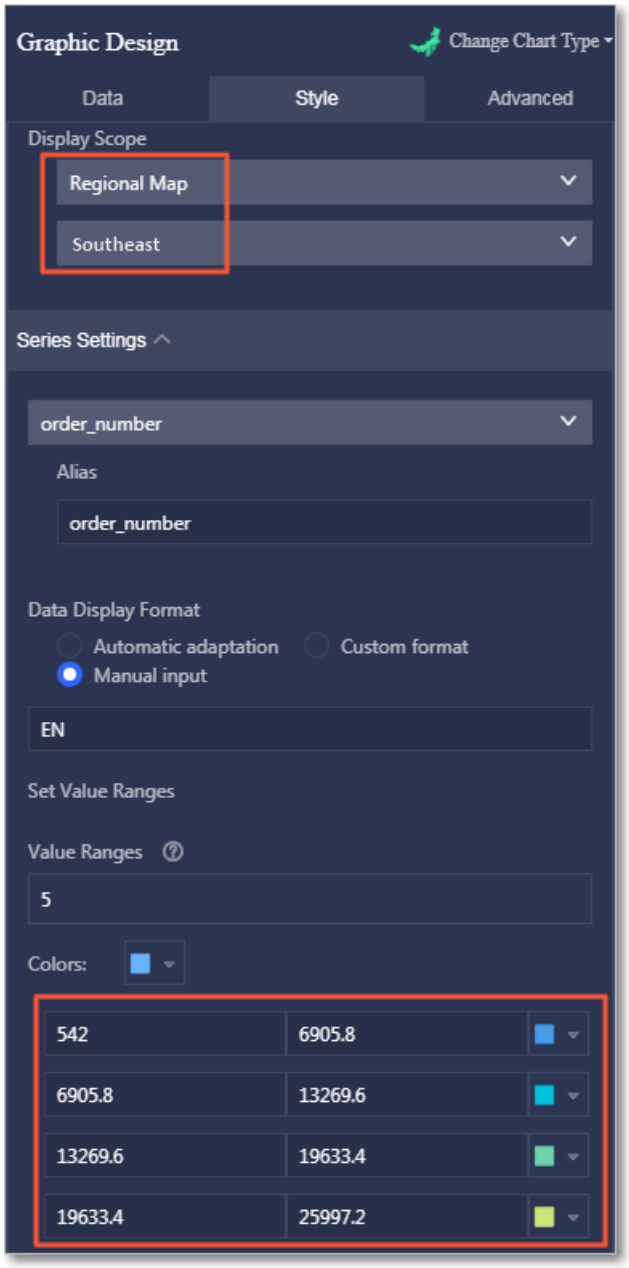
1. Log on to the Quick BI console.

2. In the left-side navigation pane of the **Workspace** tab, click **Datasets** to go to the Datasets page.
3. Find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column, select Standard in the dialog box that appears, and click **OK**.
4. On the dashboard edit page, click the Geo Map icon. A geo map is created in the display area of the dashboard.
5. On the **Data** tab, select the required measures and dimensions.

   In the Dimensions list, find and add the **province** dimension to the Geo Location (Dimensions) field. In the Measures list, find and add the **order_number** measure to the Colorscale (Measures) field, as shown in the following figure:

   > **Note** Ensure that you have converted the dimension type of province from String to Geo.

6. Click **Update**. The chart is updated.
7. On the **Style** tab, change the value ranges and saturation levels.



8. Click **Save** to save the dashboard.

If you want to delete the geo map, click the **More** icon in the upper-right corner of the map and select **Delete**.

7.4.4.13. Create a cross table

A cross table displays the summary of a field and classifies field values into different groups. Aggregate functions used in a cross table include SUM, AVG, COUNT, MAX, and MIN.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A cross table consists of rows and columns. Rows are horizontal and determined by dimensions such as province and product type. Columns are vertical and determined by measures such as order quantity and profit.

There is no limit to the number of dimensions and measures that you can include in a cross table.

The following example uses the company_sales_record dataset to describe how to use a cross table to compare the packaging, shipping costs, order quantities, and average profits of different products in multiple provinces.

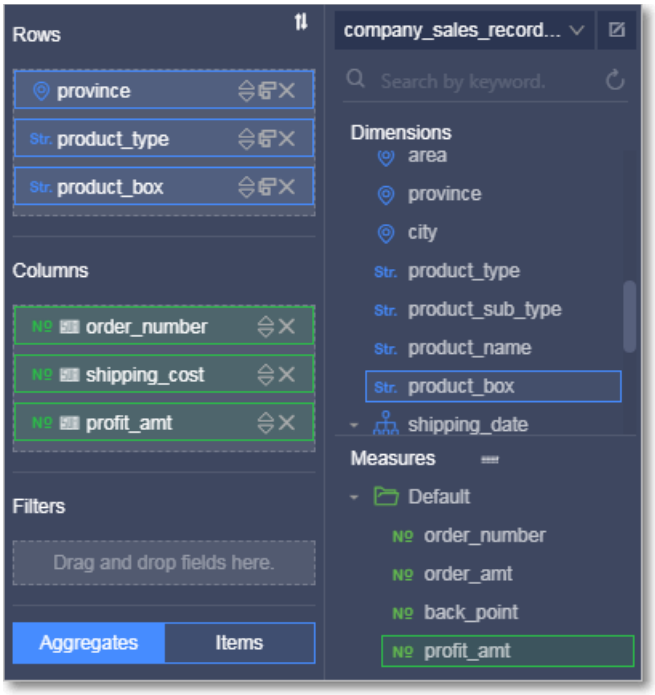
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Cross Table** icon.
5. Click the **Data** tab and select dimensions and measures.

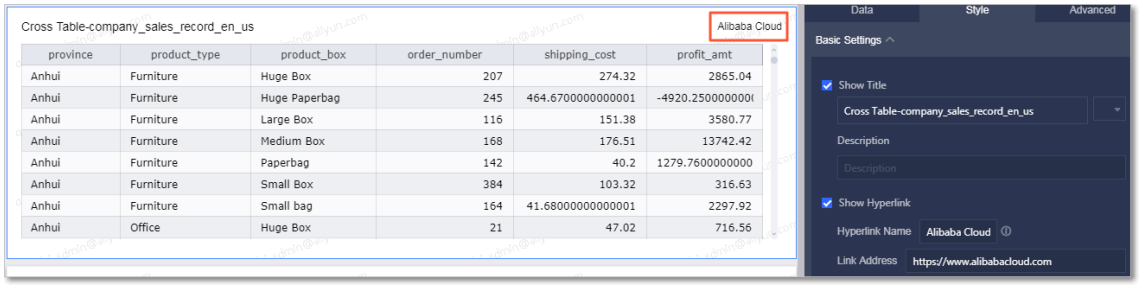
   In the Dimensions list, find and add province, product_type, and product_box to the Rows field. In the Measures list, find and add order_number, shipping_cost, and profit_amt to the Columns field, as shown in Specify fields for the cross table.

   > **Note** Ensure that you have converted the dimension type of province from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

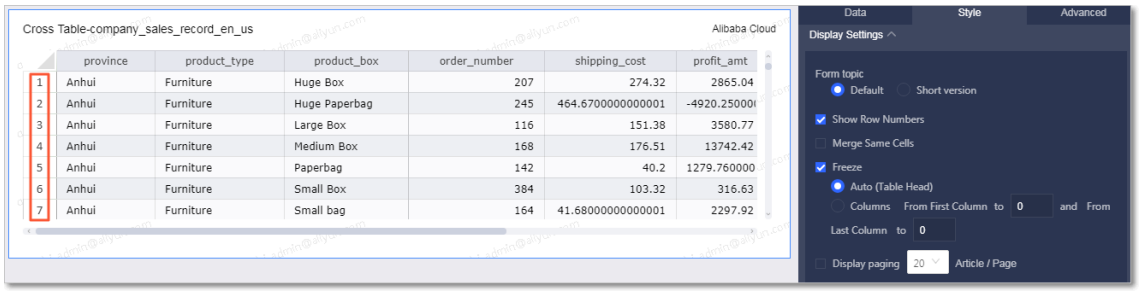
   Specify fields for the cross table



6. Click **Update**. The chart is updated.
7. Click the **Style** tab and perform the following operations:
   - In the **Basic Settings** section, specify a title and hyperlink for the cross table, as shown in the following figure.



   - In the **Display Settings** section, specify whether to show row numbers, merge cells that belong to the same category, freeze all columns by selecting Auto (Table Head), or freeze specific columns as required. After you configure the parameters, update the table. The following figure shows an updated chart.



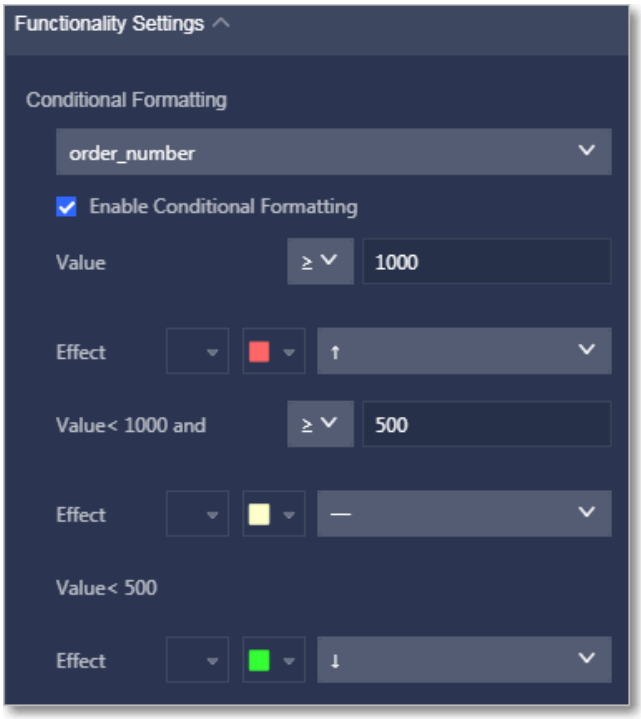
   - In the **Functionality Settings** section, set conditional formatting and sort columns.

     Conditional formatting

- Select the target field and select the **Enable Conditional Formatting** check box to enable conditional formatting. To disable conditional formatting, clear the check box, as shown in the following figure.

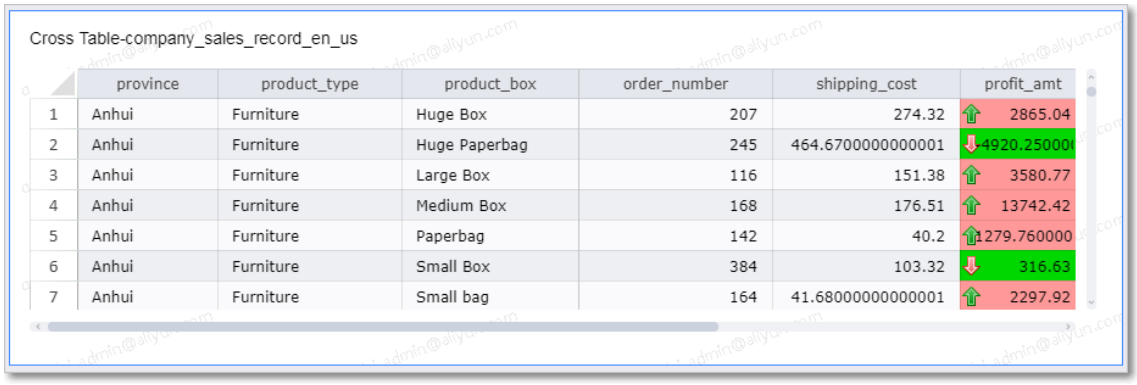
- Click the drop-down icon and select another field, as shown in the following figure.

- Specify a value for the Value parameter, click the Color icons next to the Effect parameter, and select colors to mark the values, as shown in the following figure.



In this example, the **profit_amt** field is selected for conditional formatting. The conditional formatting rules are as follows:

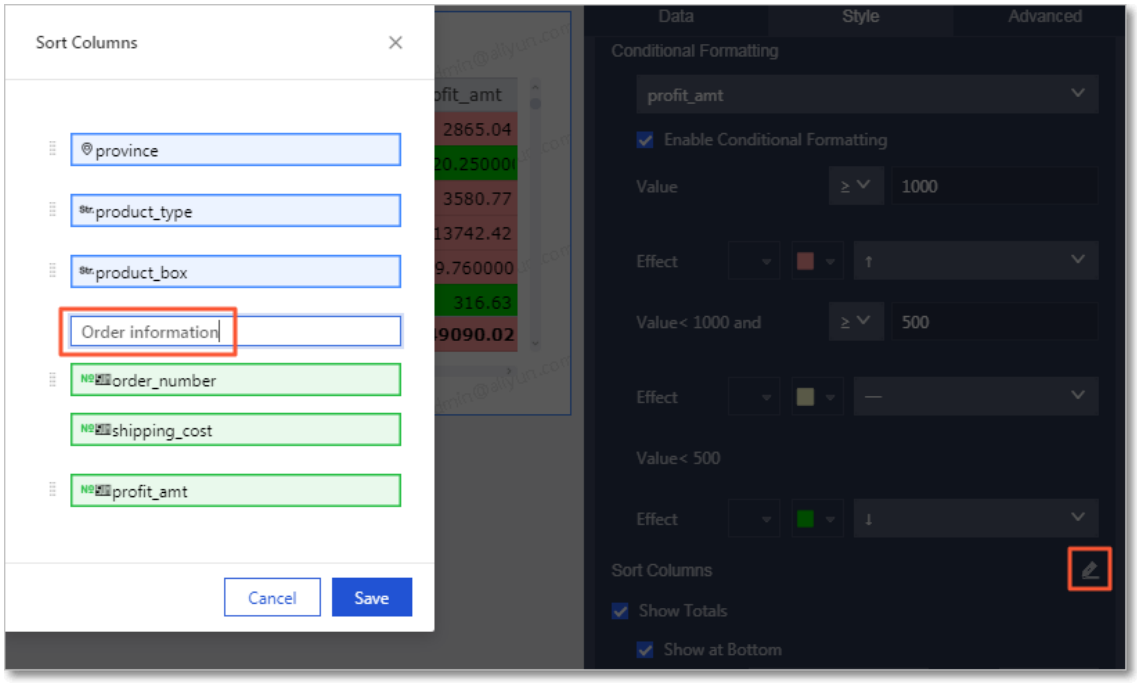
- Cells with values greater than 1000 are highlighted in red and are marked with a green upward arrow.
- Cells with values from 500 to 1000 are highlighted in gray and are marked with an orange hyphen.
- Cells with values less than 500 are highlighted in green and are marked with a red downward arrow.

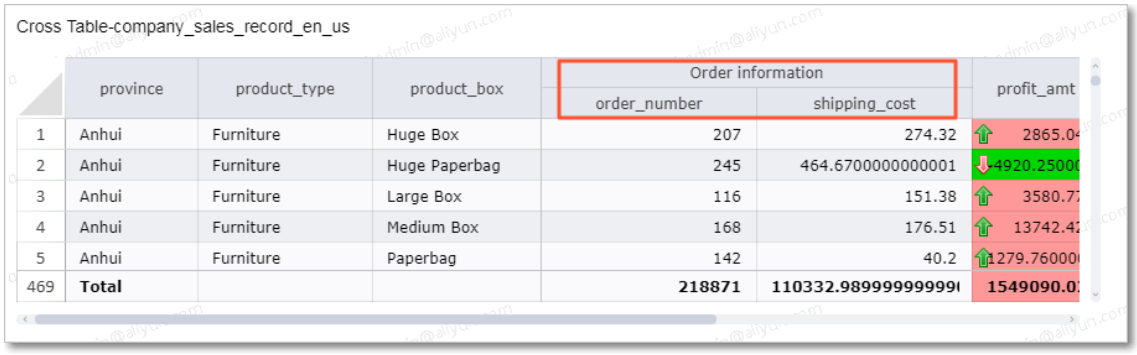

Cross Table-company_sales_record_en_us

	province	product_type	product_box	order_number	shipping_cost	profit_amt
1	Anhui	Furniture	Huge Box	207	274.32	2865.04
2	Anhui	Furniture	Huge Paperbag	245	464.6700000000001	-4920.25000
3	Anhui	Furniture	Large Box	116	151.38	3580.77
4	Anhui	Furniture	Medium Box	168	176.51	13742.42
5	Anhui	Furniture	Paperbag	142	40.2	1279.760000
6	Anhui	Furniture	Small Box	384	103.32	316.63
7	Anhui	Furniture	Small bag	164	41.68000000000001	2297.92

Sort columns

This function allows you to sort columns into different groups. You must name the groups. Otherwise, only the column order is changed.

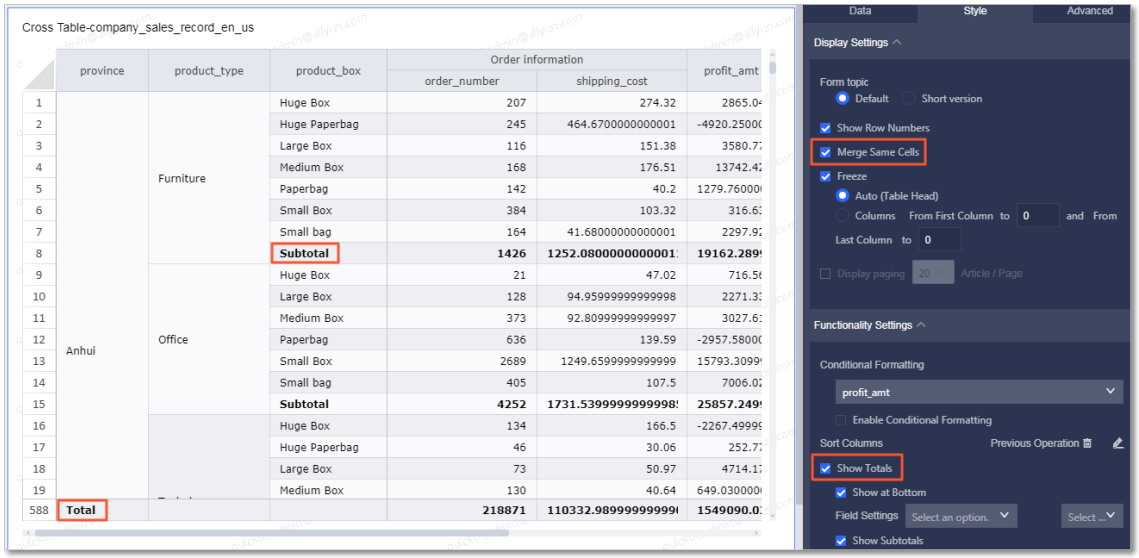



Cross Table-company_sales_record_en_us

	province	product_type	product_box	Order information		profit_amt
				order_number	shipping_cost	
1	Anhui	Furniture	Huge Box	207	274.32	2865.04
2	Anhui	Furniture	Huge Paperbag	245	464.6700000000001	-4920.25000
3	Anhui	Furniture	Large Box	116	151.38	3580.77
4	Anhui	Furniture	Medium Box	168	176.51	13742.42
5	Anhui	Furniture	Paperbag	142	40.2	1279.76000
469	**Total**			**218871**	**110332.98999999999**	**1549090.0**

- Select **Show Totals** to obtain the subtotal or total amount of data items. You can also select an aggregate function, as shown in the following figure.

> **Note** If you want to obtain the subtotal of grouped data items, you must select **Merge Same Cells** in the **Display Settings** section.



- In the **Series Settings** section, rename fields, set the alignment mode, and specify the number of records displayed on a single page.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.14. Create a pivot table

A pivot table displays aggregates of variables and allows you to analyze data in a tree structure. One variable defines the values in the header row while the other variable defines the values in the header column. Aggregate functions include SUM, AVG, COUNT, MAX, and MIN.

Similar to a cross table, a pivot table consists of rows and columns. You can specify an unlimited number of dimensions, such as province and product type, to determine rows and an unlimited number of measures, such as order quantity and profit to determine columns.

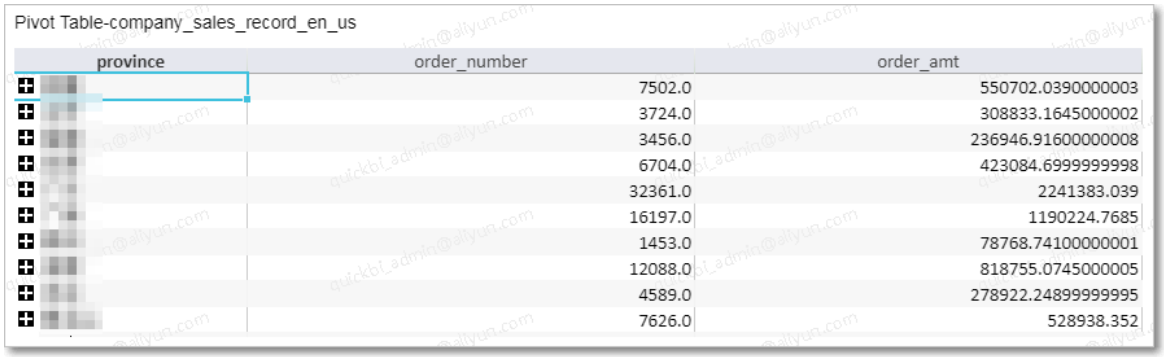
The following example uses the company_sales_record dataset to describe how to use a pivot table to show the packaging, order quantities, and order prices of different products in different provinces.

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Pivot Table** icon. A pivot table appears in the display area of the dashboard.
5. Click the **Data** tab and select the required dimensions and measures.

   In the Dimensions list, find and add **province**, **product_type** and **product_box** to the Rows (Dim.) field. In the Measures list, find and add **order_number** and **order_amt** to the Values (Mea.) field.

> **Note** Ensure that you have converted the dimension type of province from String to Geo.

6. Click **Update**. The chart is updated.



7. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.15. Create a gauge

Similar to a dashboard in a car, a gauge shows the range of a specific metric. You can view the progress of the current task or determine whether a metric exceeds its range in a gauge. For example, you can use a gauge to show the inventory status of a commodity, which helps you replenish the inventory in a timely manner.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A gauge consists of the pointer angle and tooltip. The pointer angle and tooltip are determined by a measure, such as profit or discount.

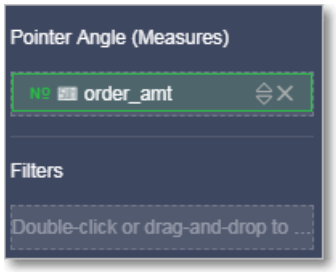
For each gauge, you can specify only one measure for the pointer angle and tooltip.

The following example uses the company_sales_record dataset to describe how to use a gauge to show order prices.

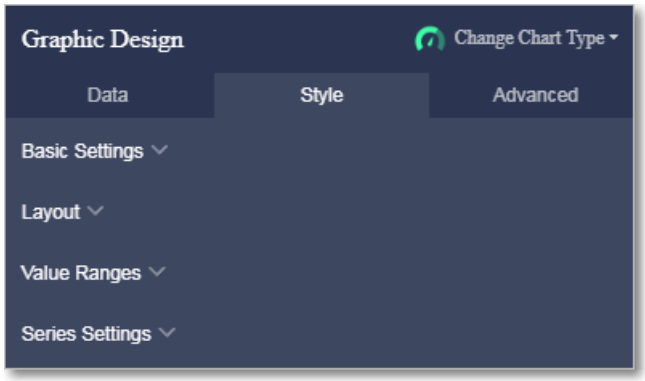
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Gauge** icon.
5. Click the **Data** tab and select the required measure.

In the Measures list, find and add order_amt to the Pointer Angle (Mea.) or Tooltip (Mea.) field.



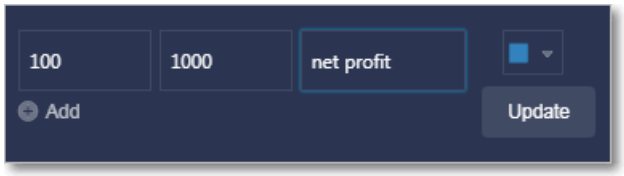
6. Click **Update**. The chart is updated.
7. Click the **Style** tab, and set the title, layout, and show or hide legend and tick marks.



8. Click **Add** in the Value Ranges section, and specify the start value and end value.

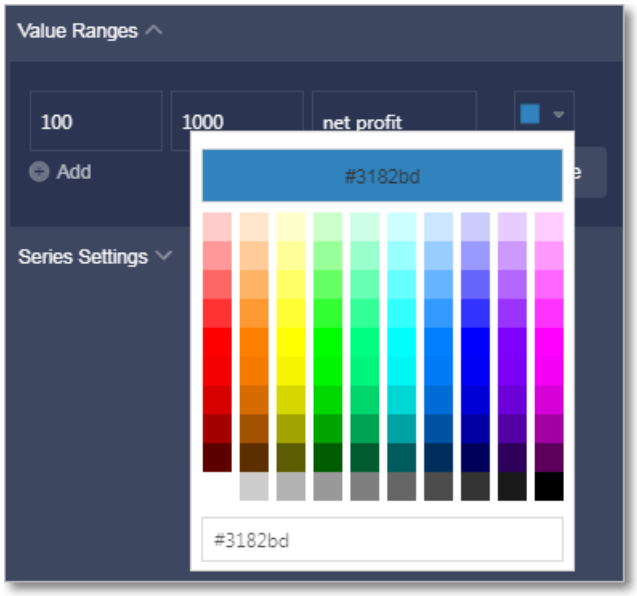
   For example, you can set the start value to 100, the end value to 1000, and the range title to Net Profit, as shown in Set a value range.

   Set a value range



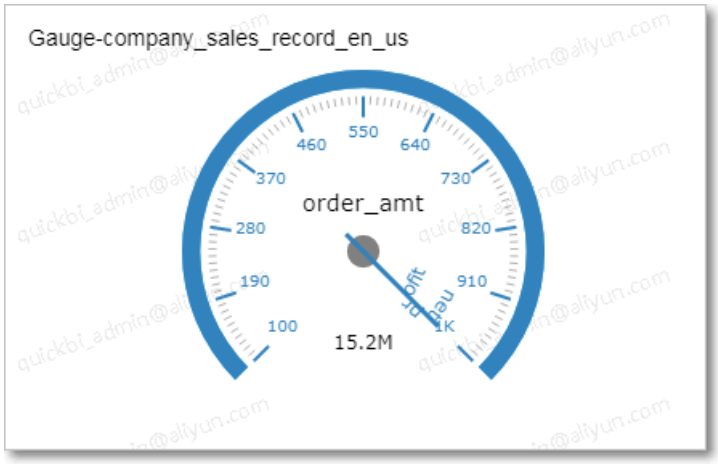
9. Click the Color icon and select a color for the value range, as shown in Change colors for value ranges.

   Change colors for value ranges



10. Click **Update**. The chart is updated, as shown in Gauge.

    Gauge



11. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
12. Click **OK** to save the dashboard.

    If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.16. Create a radar chart

A radar chart shows numbers or ratios obtained from analysis. It allows you to learn about the changes and trends of data metrics. For example, you can use a radar chart to show the sales volumes in different regions.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A radar chart consists of labels and label lengths. Labels are determined by dimensions, such as product type. Label lengths are determined by measures, such as shipping cost.

You can specify one or two dimensions for labels. we recommend that the number of dimension values be greater than or equal to three and less than or equal to 12. You can specify at least one measure for label lengths.

The following example uses the company_sales_record dataset to describe how to use a radar chart to compare the order quantities and order price in different regions.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Radar Chart** icon.
5. Click the **Data** tab and select the required dimensions and measures.

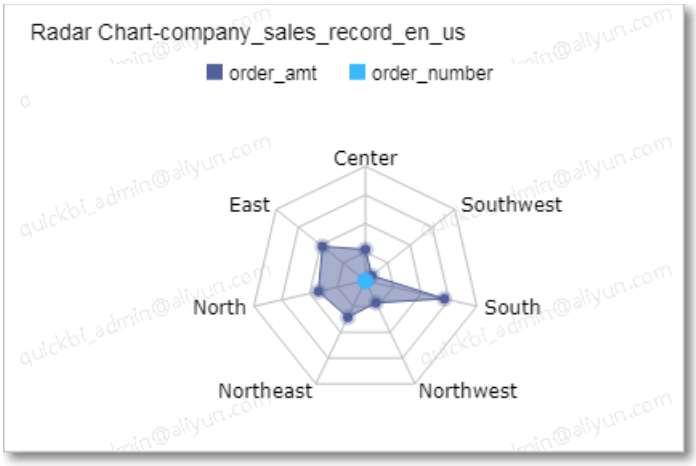
   In the Dimensions list, find and add area to the Labels (Dim.) field. In the Measures list, find and add order_number and order_amt to the Length (Mea.) field, as shown in Specify fields for the radar chart.

   Specify fields for the radar chart

6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title, layout, and legend position of the radar chart, as shown in Radar chart.

   Radar chart



8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.17. Create a scatter chart

A scatter chart displays data distribution and aggregation.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A scatter chart consists of x-axes and only one y-axis. You can specify only one dimension, such as product type, for the color legend in a scatter chart. You can specify measures for x-axes and y-axis.

There can be a maximum of 1,000 values in the Dimensions list.

You can specify one to three measures for the x-axes.

You can specify only one measure for the y-axis.

The following example uses the company_sales_record dataset to describe how to use a scatter chart to compare the unit prices and order quantities of different products.

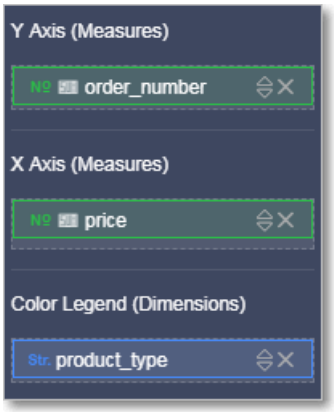
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Scatter Chart** icon.

5. Click the **Data** tab and select the required dimension and measures.

   In the Dimensions list, find and add product_type to the Color Legend (Dim.) field. In the Measures list, find and add price to the X Axis (Mea.) field and order_number to the Y Axis (Mea.) field, as shown in Specify fields for the scatter chart.

   Specify fields for the scatter chart

6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title, layout, and legend position of the scatter chart, as shown in Scatter chart.

   Scatter chart

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.18. Create a bubble chart

A bubble chart displays data distribution and aggregation by placing proportionally sized bubbles in corresponding locations.

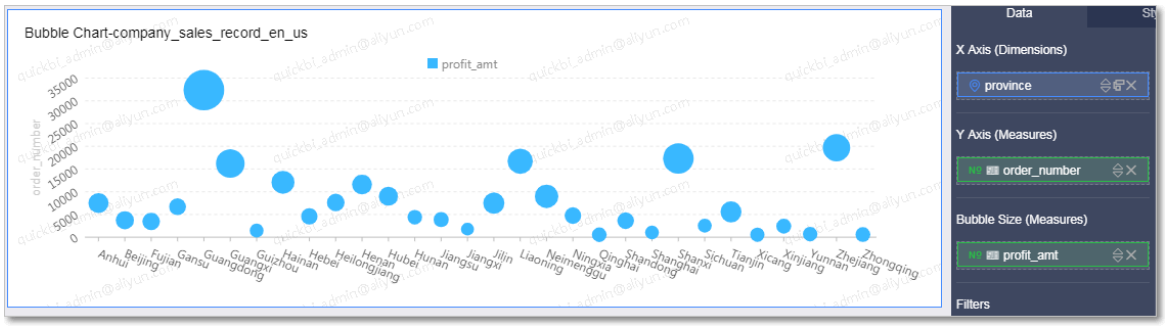
Notes

A bubble chart consists of an x-axis, a y-axis, and bubbles of different sizes. You can specify only one dimension for the x-axis, one measure for the y-axis, and one measure for the bubble size.

The following example uses the company_sales_record dataset to describe how to use a bubble chart to compare the order quantities and profits in different provinces.

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the Bubble Chart icon. A bubble chart appears in the display area of the dashboard.
5. Click the **Data** tab and select the required dimension and measures.

   In the Dimensions list, find and add **province** to the X Axis (Dim.) field. In the Measures list, find and add **order_number** to the Y Axis (Mea.) field and **profit_amt** to the Bubble Size (Mea.) field. Click the **Style** tab and change the title, layout, and legend position of the bubble chart.
6. Click **Update**. The chart is updated.



7. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.

If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.19. Create a funnel chart

A funnel chart is suitable for analyzing standard, long-running, and multi-flow business processes. By comparing business data from different stages, funnel charts allow you to explore and analyze problems. You can also use a funnel chart to show the conversion rates between the stages of a business process, such as the percentage of visitors that become paying customers for a shopping website. Funnel charts are ideal for business process analysis.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A funnel chart consists of tier labels and tier areas. Tier labels are determined by a dimension, such as area. Tier areas are determined by a measure, such as order price.

You can specify only one dimension for tier labels and only one measure for tier areas.

The following example uses the company_sales_record dataset to describe how to use a funnel chart to compare the order price in different regions.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Funnel Chart** icon.
5. Click the **Data** tab and select the required dimension and measure.

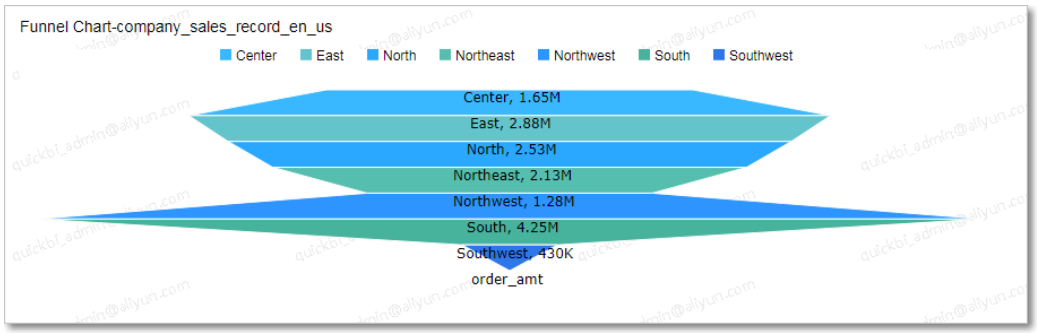
   In the Dimensions list, find and add area to the Tier Labels (Dim.) field. In the Measures list, find and add order_amt to the Tier Area (Mea.) field, as shown in Specify fields for the funnel chart.

   Specify fields for the funnel chart

6. Click **Update**. The chart is updated.
7. Click the **Style** tab and configure the title and legend position of the funnel chart, as shown in Funnel chart.

   Funnel chart

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.20. Create a kanban

A kanban provides an overview of data such as sales performance. It presents the sales status and management situation, based on which you can quickly formulate solutions, identify issues, and perform troubleshooting.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A kanban consists of metrics and labels. Labels are determined by a dimension, such as area. Metrics are determined by measures, such as order quantity and order price.

You can specify only one dimension for labels and one to ten measures for metrics.

The following example uses the company_sales_record dataset to describe how to use a kanban to compare the order quantities, order price, shipping costs, and profits in different provinces.

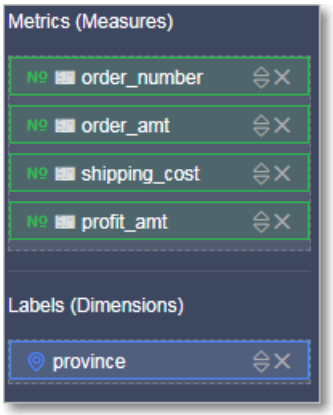
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Kanban** icon.
5. Click the **Data** tab and select the required dimension and measures.

   In the Dimensions list, find and add province to the Labels (Dim.) field. In the Measures list, find and add order_number, order_amt, shipping_cost, and profit_amt to the Metrics (Mea.) field, as shown in Specify fields for the kanban.

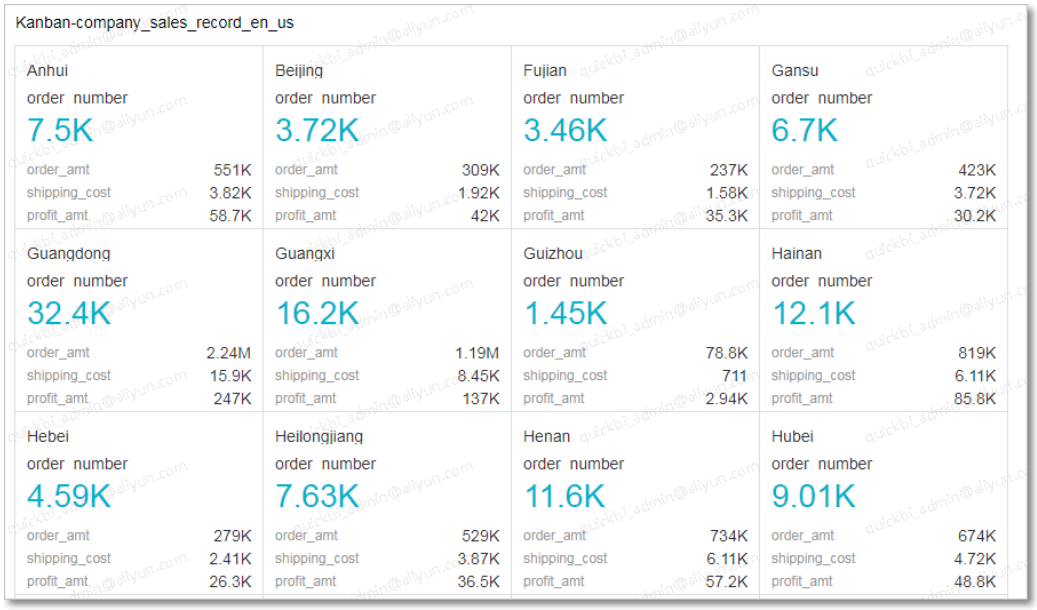
   > **Note** Ensure that you have converted the dimension type of province from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

   Specify fields for the kanban

6. Click **Update**. The chart is updated.
7. Click the **Style** tab, and set the Columns Allowed field to 3, as shown in Kanban.

Kanban



8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.21. Create a trend indicator chart

A trend indicator chart displays multiple indicators over a period of time. By default, the latest data record of each indicator is displayed.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

You can specify only one date dimension for Date (Dimensions) and must specify at least one measure for Indicator (Measures).

The following example uses the company_sales_record dataset to describe how to use a trend indicator chart to compare the order quantities, order price, shipping costs, and profits in different provinces in a specified period of time.

Create a trend indicator chart

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. In the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.

4. On the dashboard edit page, click the **Trend Indicator** icon.
5. In the Dimensions list, find and add order_date(day) to the **Date (Dimensions)** field. In the Measures list, find and add order_number, order_amt, shipping_cost, and profit_amt to the **Indicator (Measures)** field.
6. Click **Update**. The chart is updated.
7. Click the **Style** tab. Configure parameters in the **Basic Information**, **Chart Settings**, **Indicator Settings**, **Functionality Settings**, and **Series Settings** sections.
8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon ( ) in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

1. In the **Basic Information** section, configure Title, Description, Show Link, and Background.

   > **Note**
   > - This example uses light color as the background color.
   > - If you want to redirect to a report or an external page, select Show Link and specify Link Text and Link Address.

2. In the **Chart Settings** section, select Show Trend by Date, and specify a trend chart style, preview mode, and layout for multiple indicators.
   - If you select Single Choice for Select Preview Mode, the trend chart displays the trend of only one measure indicator. If you select Multiple Choice for Select Preview Mode, the trend chart displays the trend of multiple measure indicators.
   - If you select Displayed in One Line for Layout for Multiple Indicators and the indicators cannot be displayed in one line, you can slide the chart to view all indicators. If you select Displayed in Multiple Lines for Layout for Multiple Indicators, indicators are displayed in multiple lines.
3. In the **Indicator Settings** section, configure Maximum Indicators Each Row. In this example, Maximum Indicators Each Row is set to 4.
4. In the **Functionality Settings** section, configure Enable Indicator Filter and conditional formatting.

   After you select Enable Indicator Filter, an indicator filter icon is displayed in the upper-right corner of the chart. You can click the icon to select the indicators that you want to display.

   Follow these steps to configure conditional formatting:

   i. Select the target measure from the Series drop-down list. Then, select **Enable Indicator Filter** and select icon themes from the drop-down list of the Tag icon.

   ii. Specify the rules for data that you want to mark out, the icon style, and font color.

5. In the **Series Settings** section, set the measure alias, description, prefix and suffix of indicator values, and data display format.
6. On the **Advanced** tab, select Display Secondary Indicator. Select the target measure indicator from the **Select Secondary Indicator** drop-down list, and set the comparison content, display content, and symbol theme.

7.4.4.22. Create a treemap

A treemap compares the proportions of metrics of an object.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A treemap displays labels in different sizes based on the measure. Labels are determined by a dimension, such as packaging. The size of each rectangle label is determined by a measure, such as shipping costs.

You can specify only one dimension for labels and one measure for sizes. The dimension must have a maximum of 12 values.

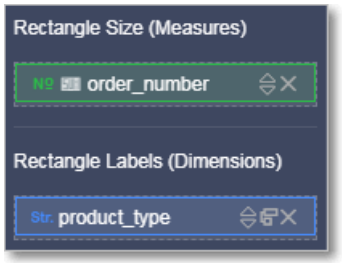
The following example uses the company_sales_record dataset to describe how to use a treemap to compare the order quantities of different products.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. In the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Treemap** icon.
5. Click the **Data** tab and select the required dimension and measures.

   In the Dimensions list, find and add product_type to the Labels (Dim.) field. In the Measures list, find and add order_number to the Size (Mea.) field, as shown in Specify fields for the treemap.

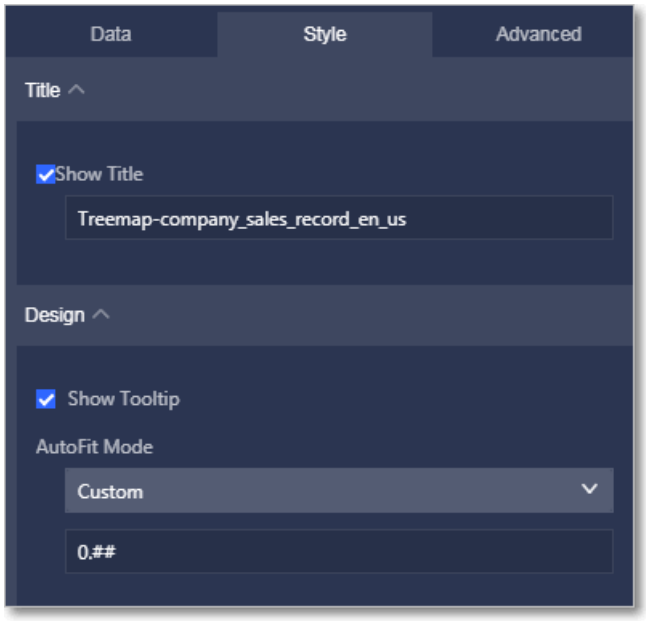
   Specify fields for the treemap

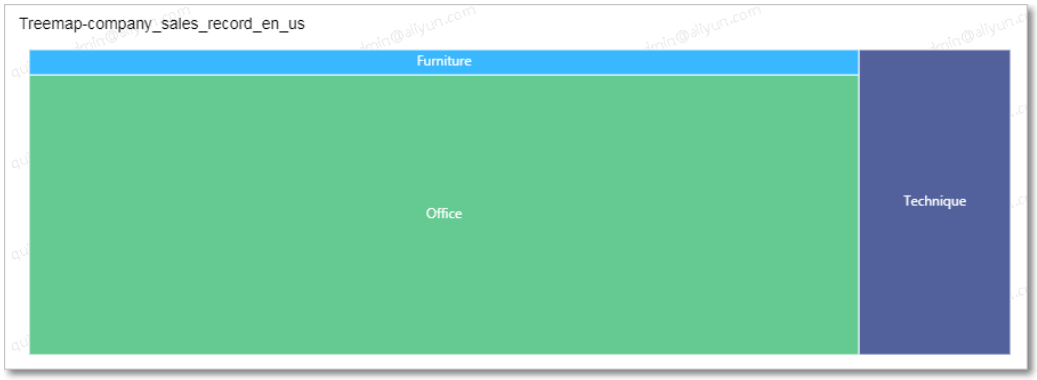
   

6. Click **Update**. The chart is updated.
7. Click the **Style** tab and configure whether to show the title and tooltip of the treemap, as

shown in Treemap.

Treemap





8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.23. Create a polar diagram

A polar diagram displays data changes over a period of time or compares metric values. It is ideal for comparing data of different objects, for example, data across different regions.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

Similar to a pie chart, a polar diagram consists of multiple slices. Slice labels are determined by a dimension, such as area or product type. The arc radius of each slice is determined by a measure, such as order quantity or order price.

You can specify only one dimension for slice labels and only one measure for arc radii.

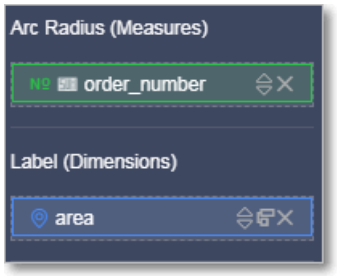
The following example uses the company_sales_record dataset to describe how to use a polar diagram to compare the order quantities in different regions. The number of regions must be greater than or equal to 3 and less than or equal to 12.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Polar Diagram** icon.
5. Click the **Data** tab and select the required dimension and measure.

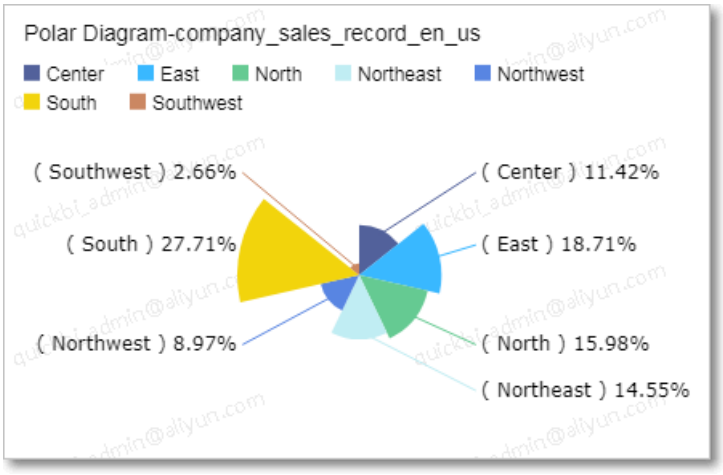
   In the Dimensions list, find and add area to the Label (Dim.) field. In the Measures list, find and add order_number to the Arc Radius (Mea.) field, as shown in Specify fields for the polar diagram.

   Specify fields for the polar diagram

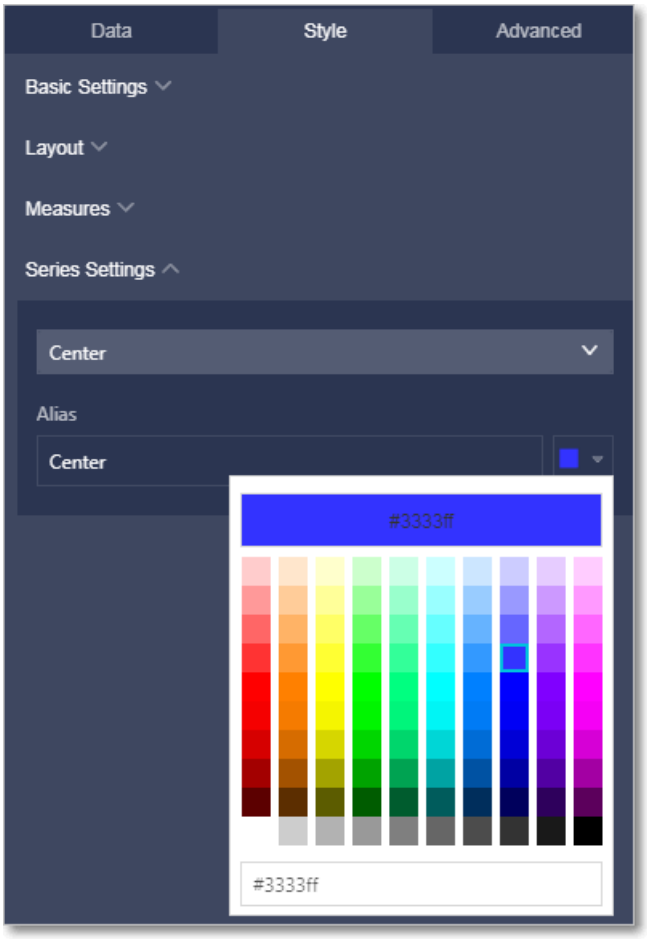
6. Click **Update**. The chart is updated.
7. Click the **Style** tab, change the title of the polar diagram, and specify the legend position, as shown in Polar diagram.

   Polar diagram



8. Click the **Style** tab and change the legend colors in the **Series Settings** section.

   Change legend colors



9. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
10. Click **OK** to save the dashboard.

    If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.24. Create a word cloud

A word cloud displays the frequency of words that appear in a dataset. It is ideal for creating user personas and user tags.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A word cloud displays words in different sizes based on their frequency of use. Words are determined by a dimension, such as customer name or product type. The size of each word is determined by a measure, such as profit or unit price.

You can specify only one dimension for words and only one measure for word sizes.

The following example uses the company_sales_record dataset to describe how to use a word cloud to compare the order quantities in different provinces.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Word Cloud** icon.
5. Click the **Data** tab and select the required dimension and measure.

   In the Dimensions list, find and add province to the Word (Dim.) field. In the Measures list, find and add order_number to the Word Size (Mea.) field, as shown in Specify fields for the word cloud.

   > **Note** Ensure that you have converted the dimension type of province from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

   Specify fields for the word cloud

6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title of the word cloud, as shown in Word cloud.

   Word cloud

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.25. Create a tornado-leaned funnel chart

A tornado-leaned funnel chart is the combination of a tornado chart and a funnel chart. A tornado chart compares different metrics between two objects, for example, the income difference and the education level difference between residents in two cities. A funnel chart shows the conversion rates between the stages of business processes, such as the percentage of visitors that become paying customers for a shopping website. Funnel charts are ideal for business process analysis.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A tornado-leaned funnel chart combines the features of a tornado chart and a funnel chart. Assume that you want to compare the percentage of the migrant population, employment rate, and commercial housing transactions in Beijing and Shanghai, and a conversion relationship exists between these metrics. The tornado-leaned funnel chart shows the values of the metrics for the two cities and also the conversion rates between the metrics.

If no conversion relationship exists, the tornado-leaned funnel chart functions the same as a tornado chart. If a conversion relationship exists between metrics and but one comparison subject is defined, the tornado-leaned funnel chart functions the same as a funnel chart.

A tornado-leaned funnel chart consists of comparison subjects and metrics. Comparison subjects are determined by a dimension, such as area or product type. Metrics are determined by measures, such as order quantity and order price.

You can specify only one dimension for the comparison subjects. You must specify at least one measure for metrics.

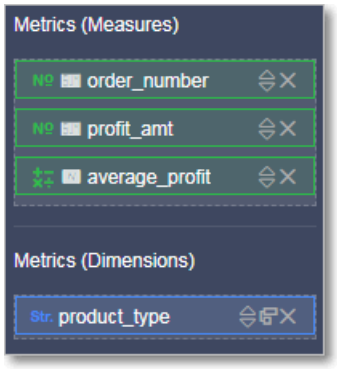
The following example uses the company_sales_record dataset to describe how to use a tornado-leaned funnel chart to compare the order quantities, profits, and average profits of different products.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Tornado-leaned Funnel Chart** icon.
5. Click the **Data** tab and select the required dimension and measures.

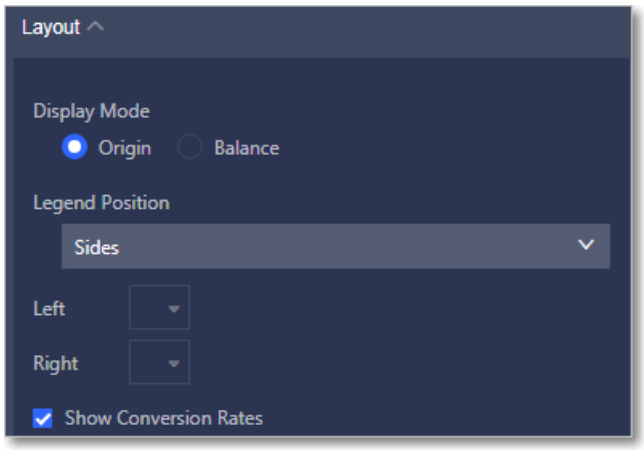
   In the Dimensions list, find and add product_type to the **Metrics (Dim.)** field. In the Measures list, find and add order_number, profit_amt, and average_profit to the **Metrics (Mea.)** field, as shown in Specify fields for the tornado-leaned funnel chart.

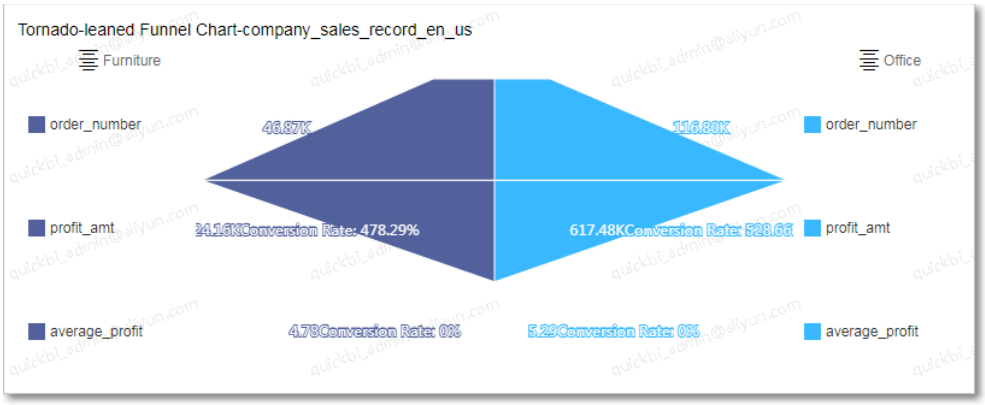
   Specify fields for the tornado-leaned funnel chart

6. Click **Update**. The chart is updated.
7. Click the **Style** tab, change the title, layout, legend position, and background color of the chart, and specify whether to show the conversion rate.
   i. Quick BI provides two layout types for tornado-leaned funnel charts. Select either of them as required.
   ii. In the **Layout** section on the Style tab, change the legend position and background color, and specify whether to show the conversion rate, as shown in Tornado-learned funnel chart.

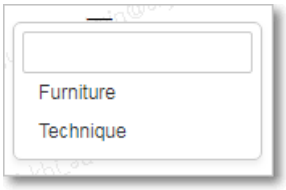
Tornado-learned funnel chart





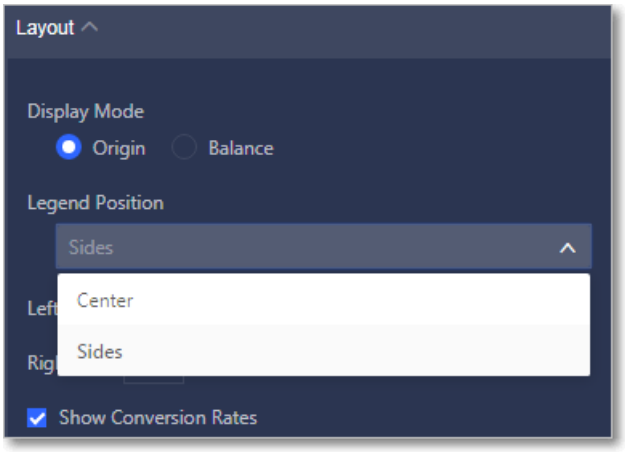
- You can move your pointer over the product type field on the chart to switch to another product, as shown in Switch to another product.

  Switch to another product

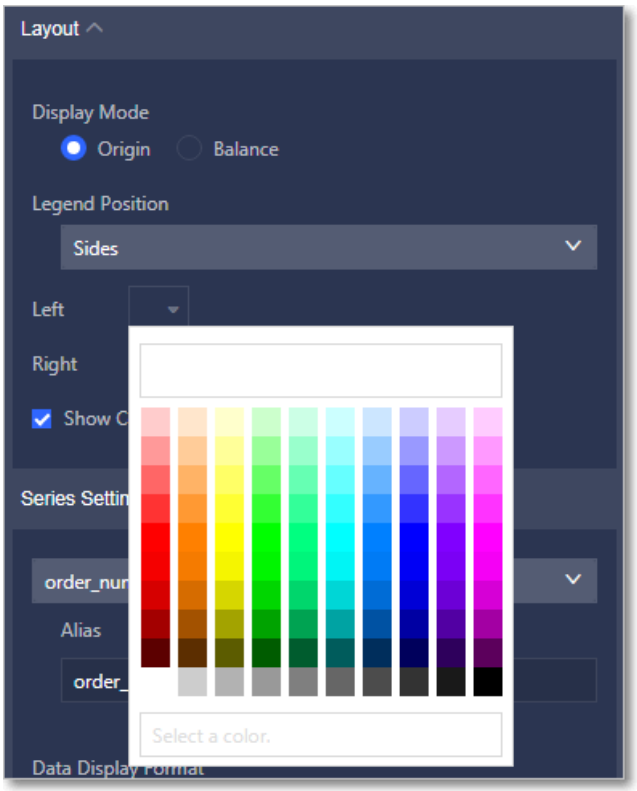
- Change the legend position, as shown in Change the legend position.

  Change the legend position

- Click the Color icon next to Left or Right and select a color from the drop-down list, as shown in Change legend colors.

  Change legend colors

- Hide or show the conversion rate, as shown in Hide or show the conversion rate.

  Hide or show the conversion rate

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.26. Create a hierarchy chart

A hierarchy chart organizes and displays hierarchical data in a tree structure. It is an implementation of the enumeration method. For example, when you view revenues of cities in a province, the relationships between the province and the cities are displayed in a hierarchical structure. Hierarchy charts are ideal for analyzing data related to organizational structures, such as the staff structure of a company or department structure of a hospital.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A hierarchy chart consists of node metrics and node labels. Node labels are determined by dimensions, such as area and product type. Node metrics are determined by measures, such as order quantity and order price.

This topic uses the following scenarios as examples to describe how to use a hierarchy chart and the filter:

- Scenario 1: Compare the order quantities of different products in the provinces of different regions.
- Scenario 2: View the average profits of different products in different municipalities.

In a hierarchy chart, you must specify at least two dimensions for node labels. Data is displayed clear if these dimensions have a hierarchical relationship. You must specify at least one measure for node metrics.

Scenario 1: Compare the order quantities of different products in the provinces of different regions

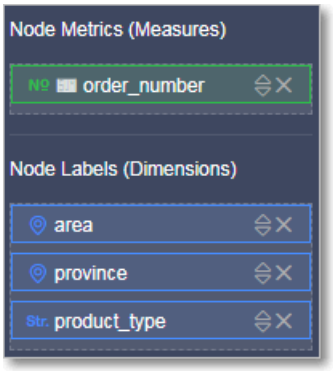
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Hierarchy Chart** icon.
5. Click the **Data** tab and select the required dimensions and measures.

   In the Dimensions list, find and add area, province, and product_type to the Node Labels (Dim.) field. The sequence in which you add these dimensions determines the hierarchical relationship in the chart. In the Measurement list, find and add order_number to the Node Metrics (Mea.) field, as shown in the following figure.

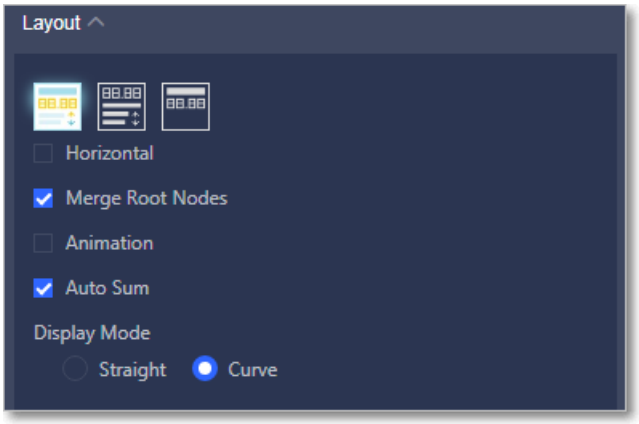
   > **Note** Ensure that you have converted the dimension type of area and province from String to Geo. For information about how to convert a dimension type, see Edit a dimension.

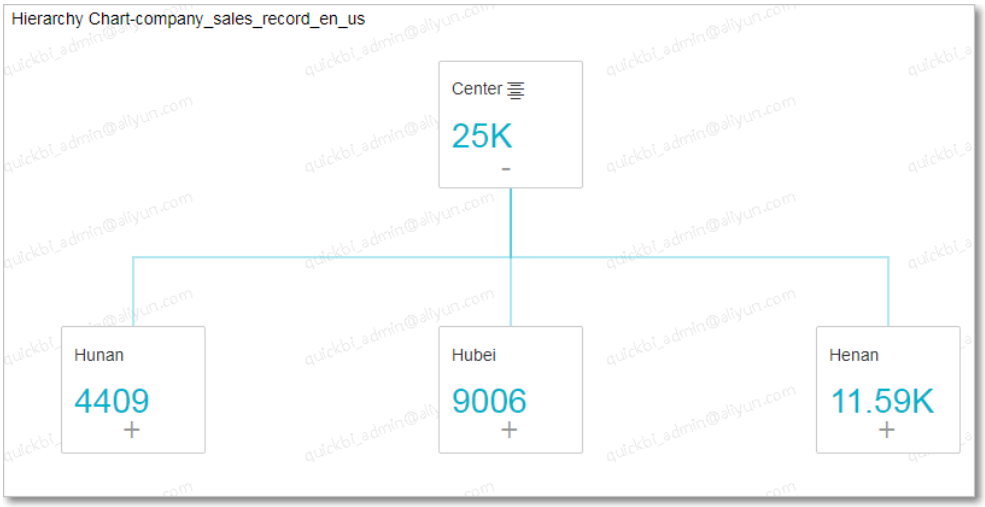
   Specify fields for the hierarchy chart



6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title, layout, and design of the chart.
   i. Quick BI provides three layout types for hierarchy charts. You can select a structure and display mode for the chart based on your business needs. The Merge Root Nodes check box is selected by default. In the following example, the **Straight** mode is selected.

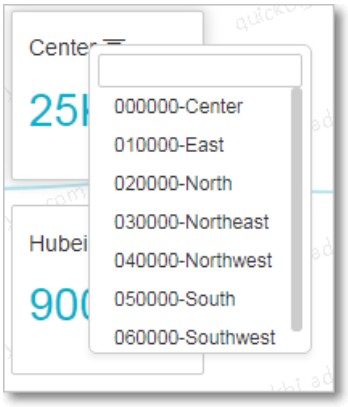
Layout





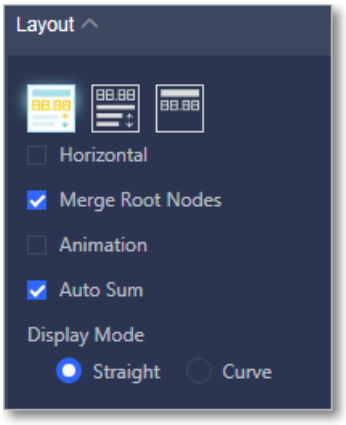
- Move your pointer over the area field in the chart and switch to another region from the drop-down list that appears, as shown in the following figure.

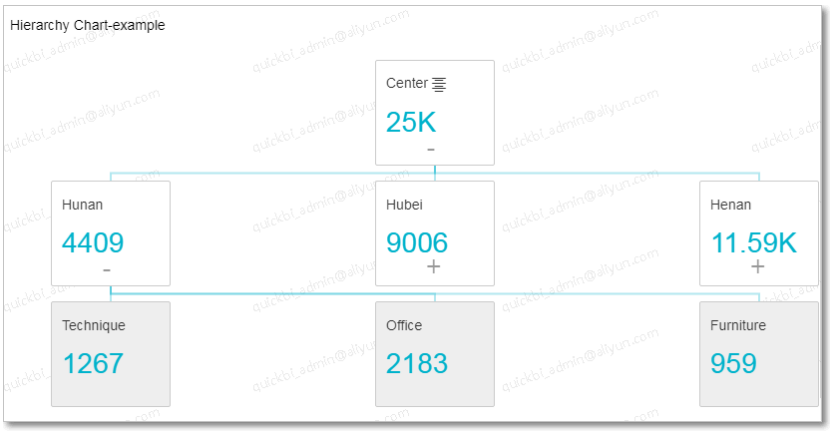
  Switch to another region

- Click the minus sign (-) or plus sign (+) to fold or unfold the child nodes.
- In the Layout section, if you select **Auto Sum**, the chart displays the total amount of each node, as shown in the following figure.

  Auto Sum

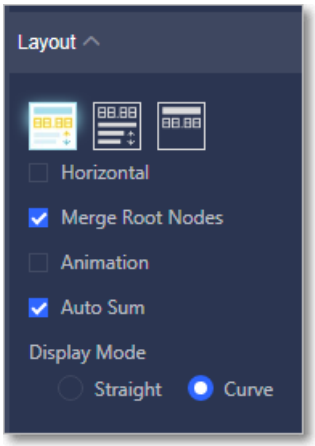
  

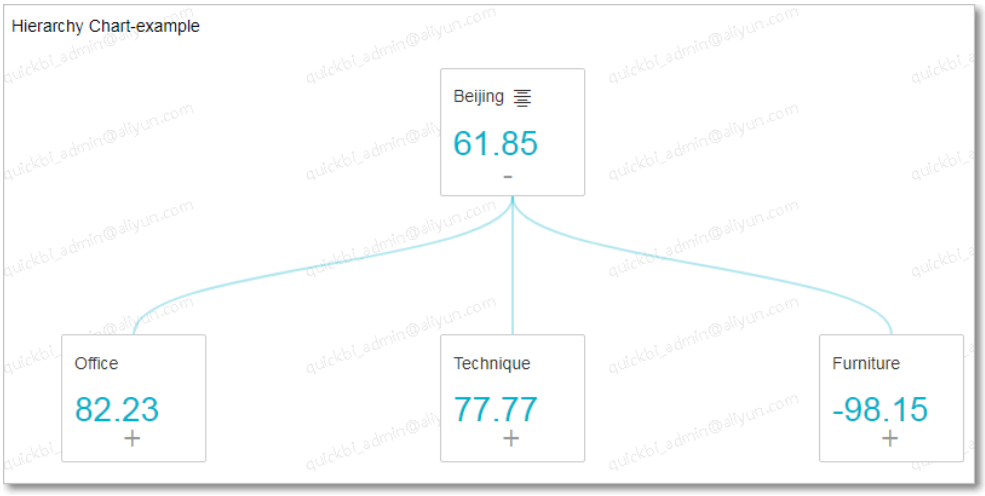
  

ii. In the Design section, you can configure the Levels field to specify the level of data to be displayed. You can also set a primary path, which is displayed in a different color in the chart. You can select the Show Filter Bar check box to add a toolbar to the chart. This allows you to edit the chart in preview mode or in the dashboard.

In the following example, the Primary Path parameter is set to order_number, the Sort parameter is set to Ascend, the Show Filter Bar check box is selected, and the **Curve** display mode is selected, as shown in the following figure.

Hierarchy chart





8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a name for the dashboard.
9. Click **OK** to save the dashboard.

   By default, the dashboard is saved to the **My Items** tab on the Dashboards page.

Scenario 2: View the average profits of different products in different municipalities

Context

Data modeling may be required in this scenario. For information about data modeling, see Add a calculated field.

Procedure

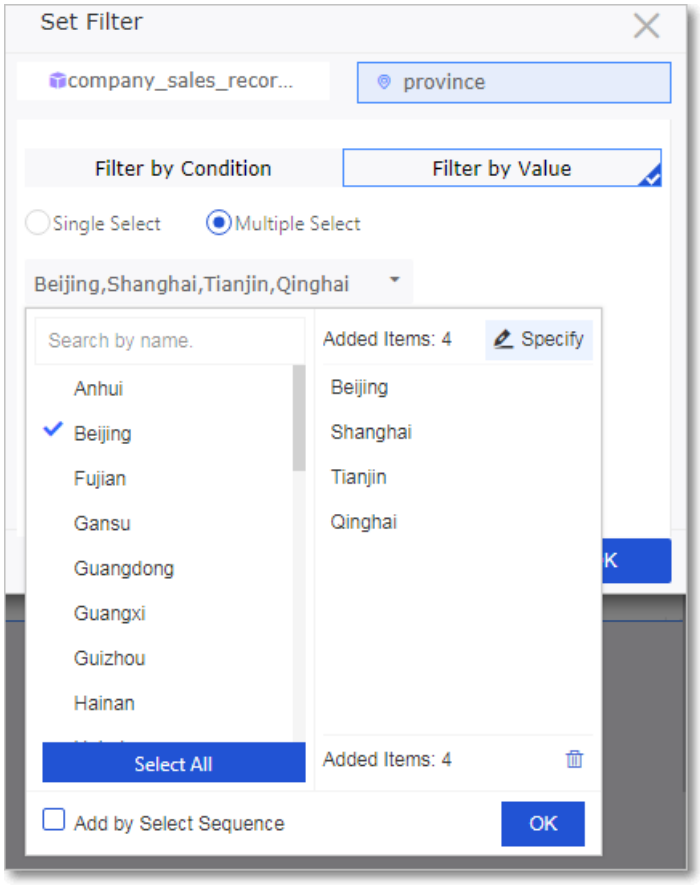
1. In the Dimensions list, find and add province to the Filters field.

   This way, you can filter municipalities.

2. Click the **Filter** icon. In the Set Filter dialog box that appears, select **Filter by Value** and **Multiple Select**, as shown in the following figure.

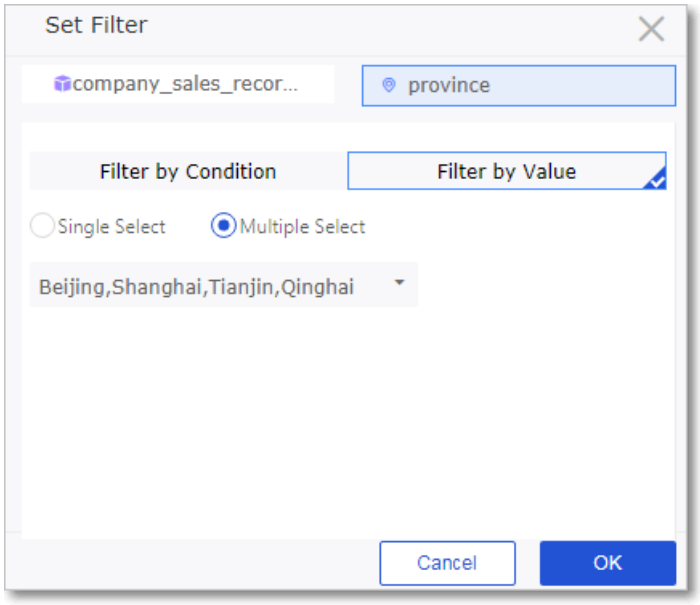
   The system automatically lists all available options of the province field in the drop-down list.

   Filter by Value



3. Select the municipalities or manually enter their names.
4. Click **OK** to set the filter condition, as shown in the following figure.

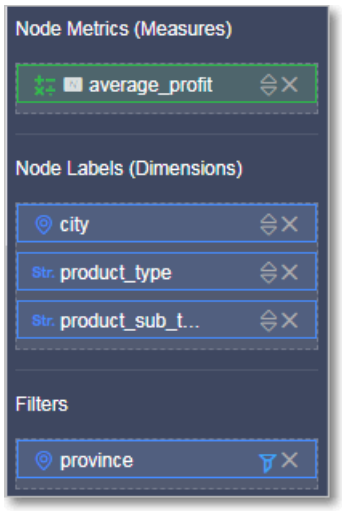
   Set the filter condition



5. In the Dimensions list, find and add city, product_type, and product_sub_type to the Node Labels (Dim.) field, as shown in the following figure.

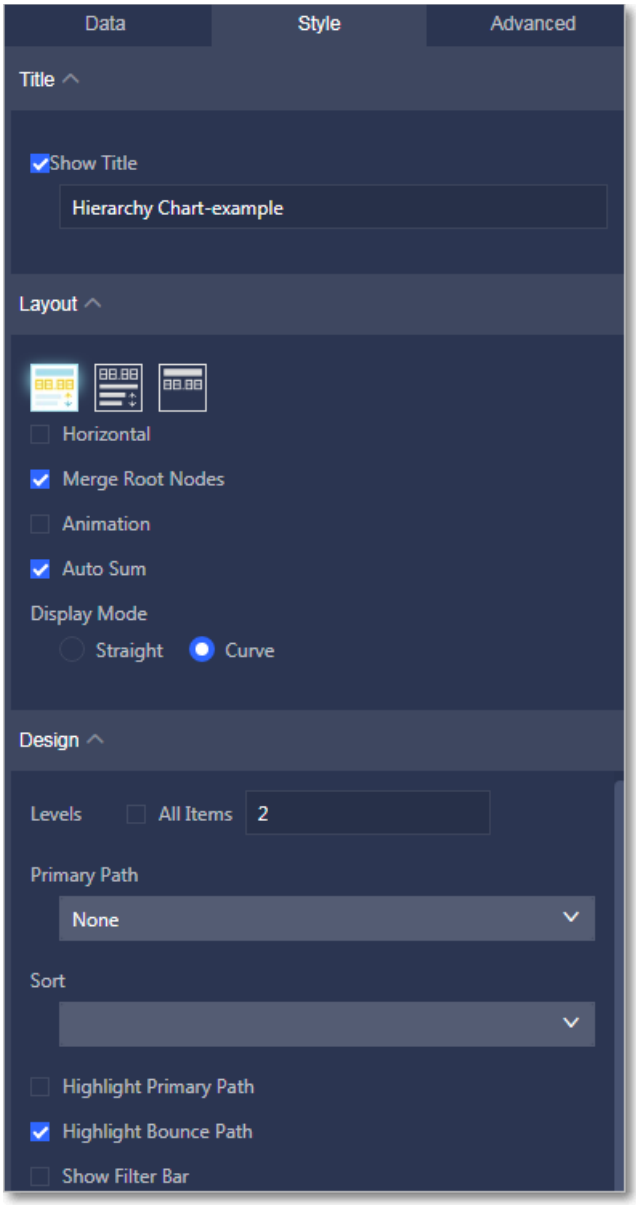
   The sequence in which you add these dimensions determines the hierarchical relationship displayed in the chart. In the Measures list, find and add average_profit to the Node Metrics (Mea.) field.

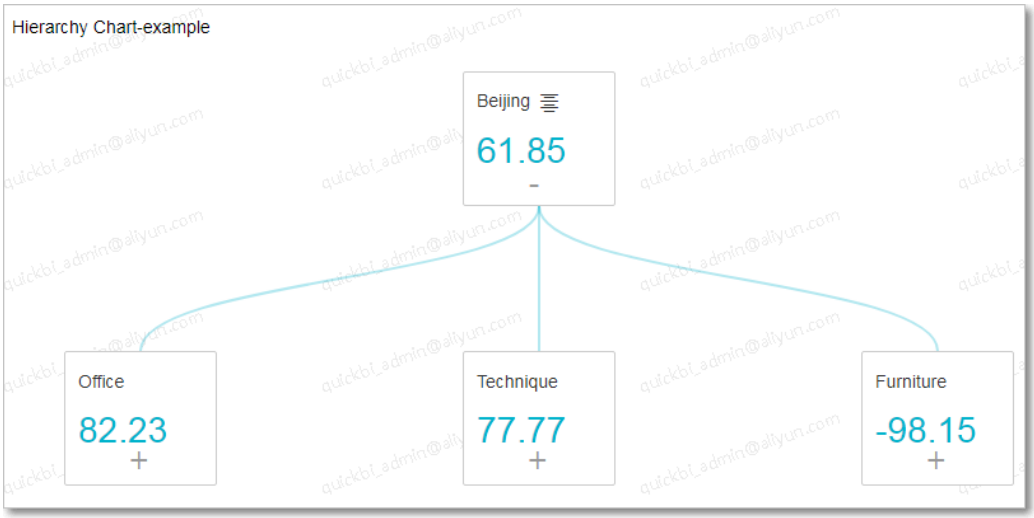
   Specify fields for the hierarchy chart



6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title, layout, and design of the chart, as shown in the following figure.

   ## Hierarchy chart





8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, enter a

name for the dashboard.

9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

7.4.4.27. Create a flow analysis chart

A flow analysis chart shows the data flow among source, central, and destination nodes.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

A flow analysis chart supports three-level dimensions, which are used to specify Central Node (Dimensions), Node Type (Dimensions), and Node Name (Dimensions). A measure is used to specify Node Indicator (Measures).

- You can specify only one dimension for each of the Central Node (Dimensions), Node Type (Dimensions), and Node Name (Dimensions) fields. The values of the Node Type (Dimensions) field include **source**, **center**, or **goal**, corresponding to source, central, and destination nodes, respectively. You can specify only one measure for the Node Indicator (Measures) field. After you select a central node, the nodes whose type is source are regarded as source nodes and the nodes whose type is goal are regarded as destination nodes. The ratio is calculated by dividing the number of unique visitors (UVs) of the source nodes by the number of UVs of the central node.
- For more information, see Flow analysis-demo table.

> **Note** The data of nodes whose type is not **source**, **center**, or **goal** is filtered out. If the name for a node is left blank, **Unknown Source** is used as the name when the node type is **source** and **Unknown destination** is used when the node type is **goal**.

Create a flow analysis chart

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the page_source_target_day dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click **OK**.
4. On the dashboard edit page, click the **Flow Analysis** icon.
5. Click the **Data** tab and select the required dimensions and measure.

   In the Dimensions list, find and add the required dimensions to the Central Node (Dimensions), Node Type (Dimensions), and Node Name (Dimensions) fields. In the Measures list, find and add the required measure to the Node Indicator (Measures) field.
6. Click **Update**. The chart is updated.
7. Click the **Style** tab and change the title and data display format of the chart.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, specify a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

1. In the **Basic Information** section, configure Title, Description, Show Link, and Background. This example uses a dark color as the background color.

   > **Note** If you want to redirect to a report or an external page, select Show Link and specify Link Text and Link Address.

2. In the **Series Settings** section, configure Alias and Data Display Format for a measure.

7.4.4.28. Create a sankey diagram

A sankey diagram is a flow diagram in which the branch width is proportional to the flow rate. It shows the data flow between two groups of values. It is ideal for visualization analysis of energy, material composition, and finance data.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

You can specify two to five dimensions, such as province and product type, for the node type. You can specify one measure, such as order quantity, at most for the node height.

The following example uses the company_sales_record dataset to describe how to use a sankey diagram to compare the order quantities and order levels of products in different regions.

Create a sankey diagram

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Sankey Diagram** icon.
5. Click the **Data** tab and select the required dimensions and measure.

   In the Dimensions list, find and add area and order_level to the **Node Type (Dimensions)** field.

   In the Measures list, find and add order_number to the **Node Height (Measures)** field.
6. Click **Update**. The chart is updated.
7. Click the **Style** tab and modify the basic information, style configuration, and series settings

of the chart.

8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, specify a name for the dashboard.
9. Click **OK** to save the dashboard.

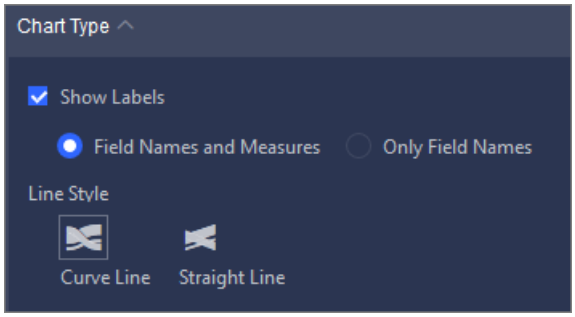
   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

1. In the **Basic Information** section, you can configure the title, whether to show the link, and the background color. In this example, **Light Color (Default)** is selected for Background.

   > **Note** If you want to redirect to a report or an external page, select **Show Link** and specify Link Text and Link Address.

2. In the **Chart Type** section, configure Show Labels and Line Style.



3. In the **Series Settings** section, configure Alias and Data Display Format for the measure. In this example, **AutoFit** is selected for Data Display Format.

7.4.4.29. Create a ranking board

A ranking board shows the ranking of the Top N objects from a specific measure in descending order. It objectively reflects the strengths of objects of the same category.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

You can specify only one dimension, such as area or product type, for the category, and only one measure, such as order quantity or profit, for indicators. A ranking board shows 20 records by default and can show up to 50 records.

The following example uses the company_sales_record dataset to describe how to use a ranking board to compare the order quantities in different areas.

Create a ranking board

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.

3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click OK.
4. On the dashboard edit page, click the **Ranking Board** icon.
5. Click the **Data** tab and select the required dimension and measure.

   In the Dimensions list, find and add **area** to the **Category (Dimensions)** field.

   In the Measures list, find and add **order_number** to the **Indicator (Measures)** field.
6. Click **Update**. The chart is updated.
7. Click the **Style** tab and modify the basic information, style configuration, and series settings of the chart.
8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, specify a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

1. In the **Basic Information** section, configure the title, color, and whether to show the link.

   > **Note** If you want to redirect to a report or an external page, select **Show Link** and specify Link Text and Link Address.
2. In the **Style Settings** section, configure Show Column Name, Theme for Top 3 Items, Data Display Format, Value Alignment, and Bar Color.
3. In the **Functionality Settings** section, configure conditional formatting.
   i. Select **Enable conditional formatting**, and select icon themes from the drop-down list of **Tag icon**.
   ii. Specify the rules for data that you want to mark out, the icon style, and font color.
4. In the **Series Settings** section, specify the alias for the measure or dimension, alignment mode, and data display format.

7.4.4.30. Create a ticker board

A ticker board displays KPI data and allows you to customize style settings such as the background color.

Prerequisites

- The Quick BI service is purchased.
- A dataset is created.

Context

Ticker boards can display only metrics (measures). The following example uses the company_sales_record dataset to describe how to use a ticker board to show order quantities.

Create a ticker board

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **Datasets**.
3. On the Datasets page that appears, find the company_sales_record dataset, click the **Create Dashboard** icon in the Actions column. In the dialog box that appears, select **Standard** and click Ok.
4. On the dashboard edit page, click the **Ticker Board** icon.
5. Click the **Data** tab. In the Measures list, find and add **order_number** to the **Indicator (Measures)** field.
6. Click **Update**. The chart is updated.
7. Click the **Style** tab and modify the basic information, style settings, function configuration, and series settings of the chart.
8. Click **Save** in the upper-right corner. In the Save Dashboard dialog box that appears, specify a name for the dashboard.
9. Click **OK** to save the dashboard.

   If you want to delete the chart, click the **More** icon in the upper-right corner of the chart and select **Delete**.

Configure parameters on the Style tab

1. In the **Basic Information** section, specify Show Title and Description, Show Link, and Background.

   You can select Light Color (Default), Color Palette, or Image URL for Background. Quick BI provides six color palettes.

   > **Note** If you want to redirect to a report or an external page, select **Show Link** and specify Link Text and Link Address.

2. In the **Style Settings** section, specify the font colors for data labels and primary indicator values and the indicator position in an indicator block.
3. In the **Functionality Settings** section, configure conditional formatting.
   i. Select **Enable conditional formatting**, and select icon themes from the drop-down list of **Tag icon**.
   ii. Specify the rules for data that you want to mark out, the icon style, and font color.
4. In the **Series Settings** section, specify the measure alias, prefix, suffix, and data display format.

7.4.5. Full Screen mode

This topic describes the basic features of the Full Screen mode.

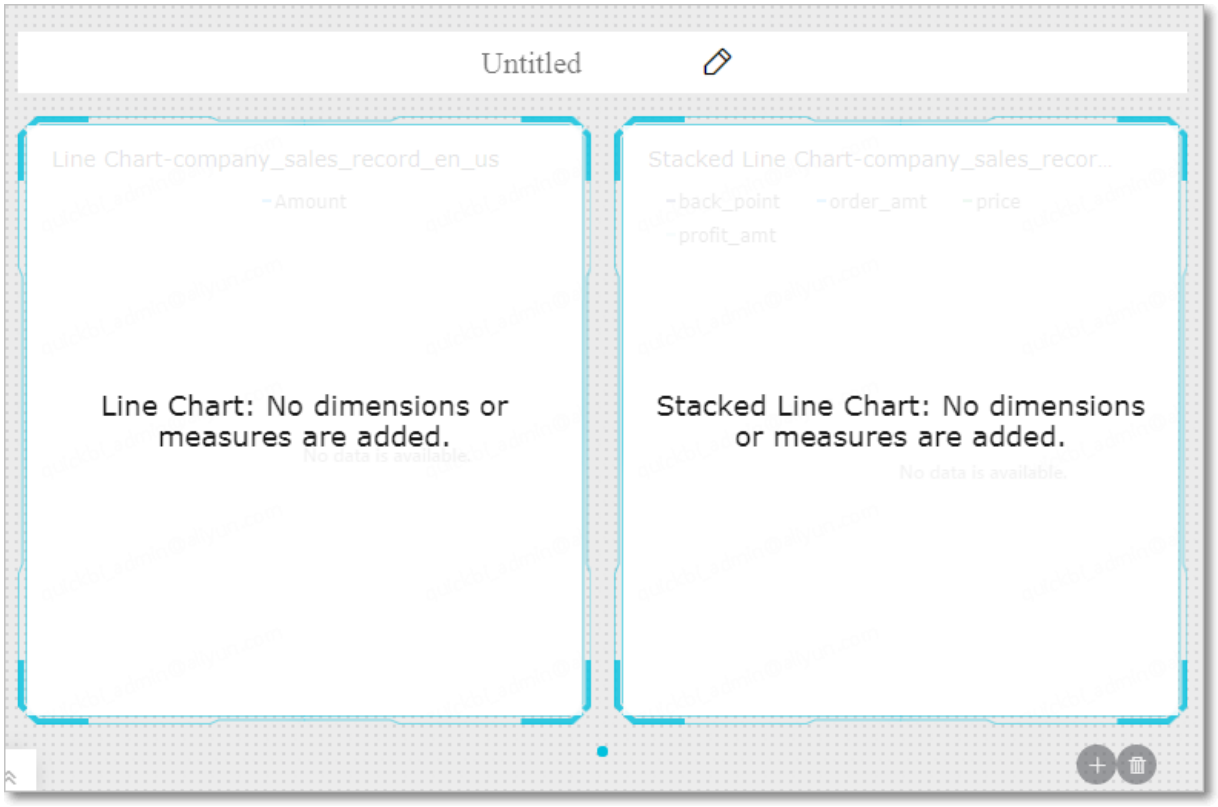
The Full Screen mode allows you to perform the following operations in the display area of a dashboard.

- Adjust chart positions
- Add a subscreen

- View chart data
- Delete a chart
- Change chart types
- Configure page settings

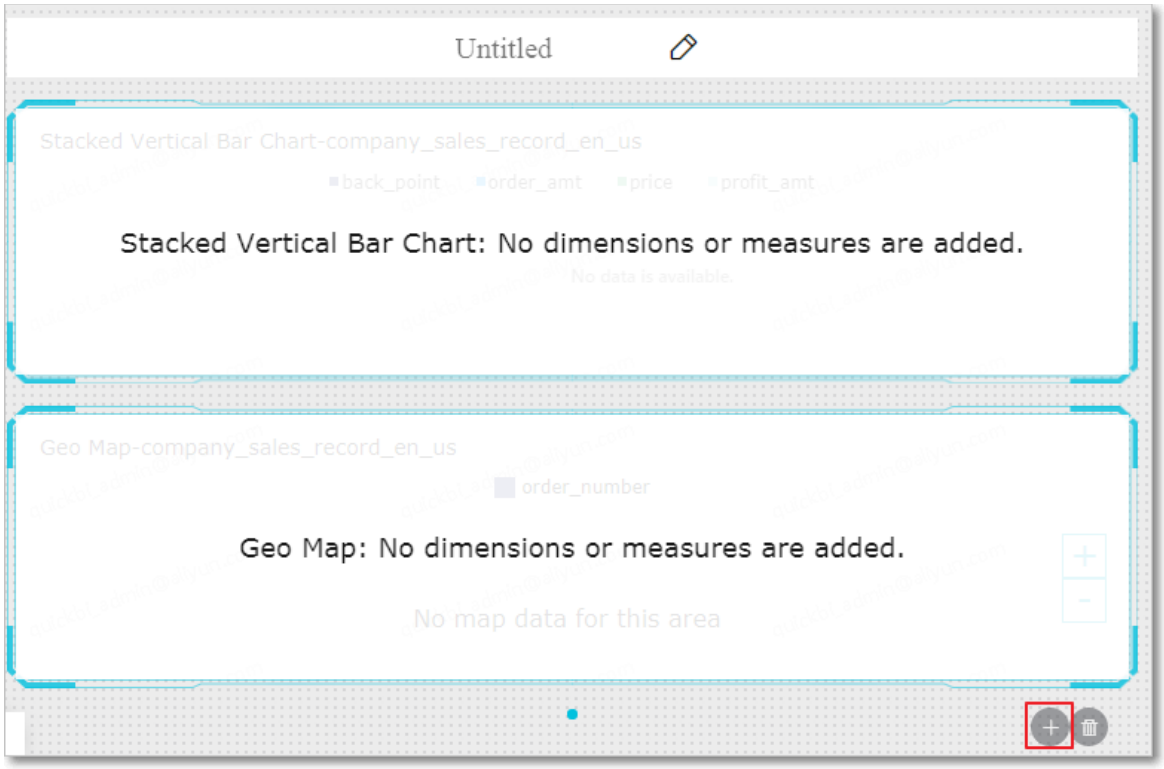
Adjust chart positions

If you create a dashboard by using the Full Screen mode and you have created only one chart, the chart covers the entire display area. If you have created multiple charts, you can click the Move icon (an arrow cross symbol), and drag the chart to the target position.

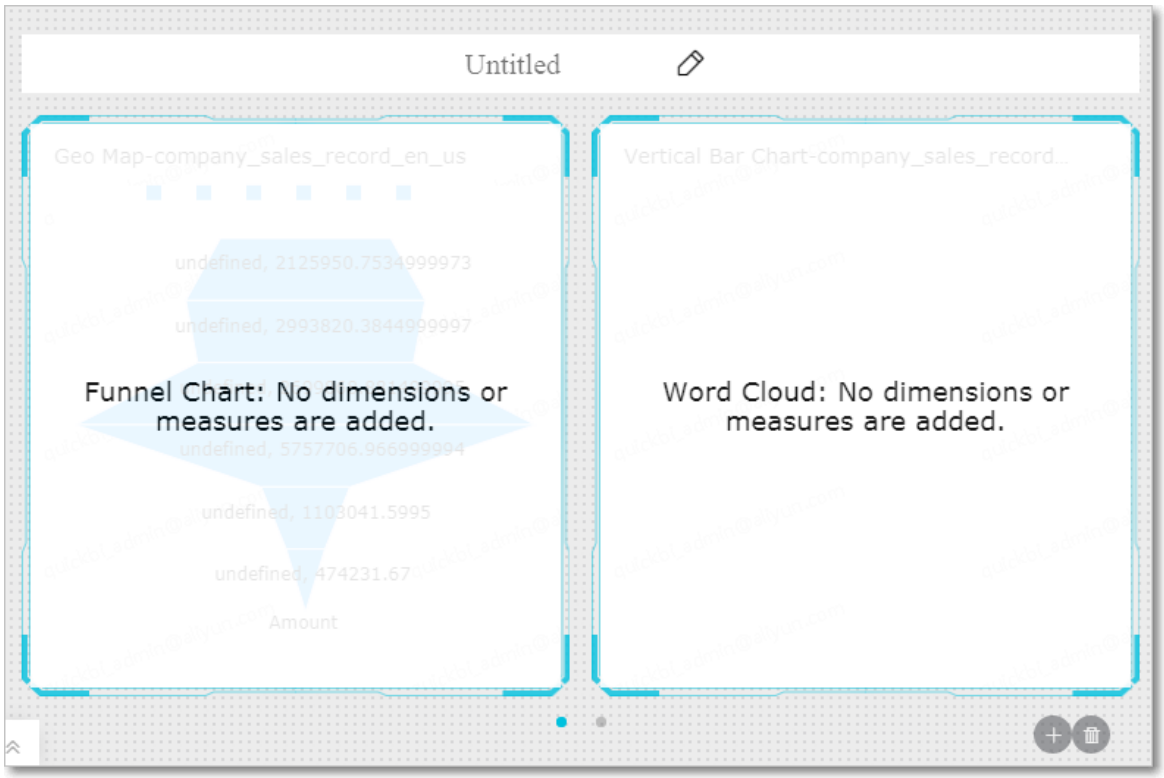


Add a subscreen

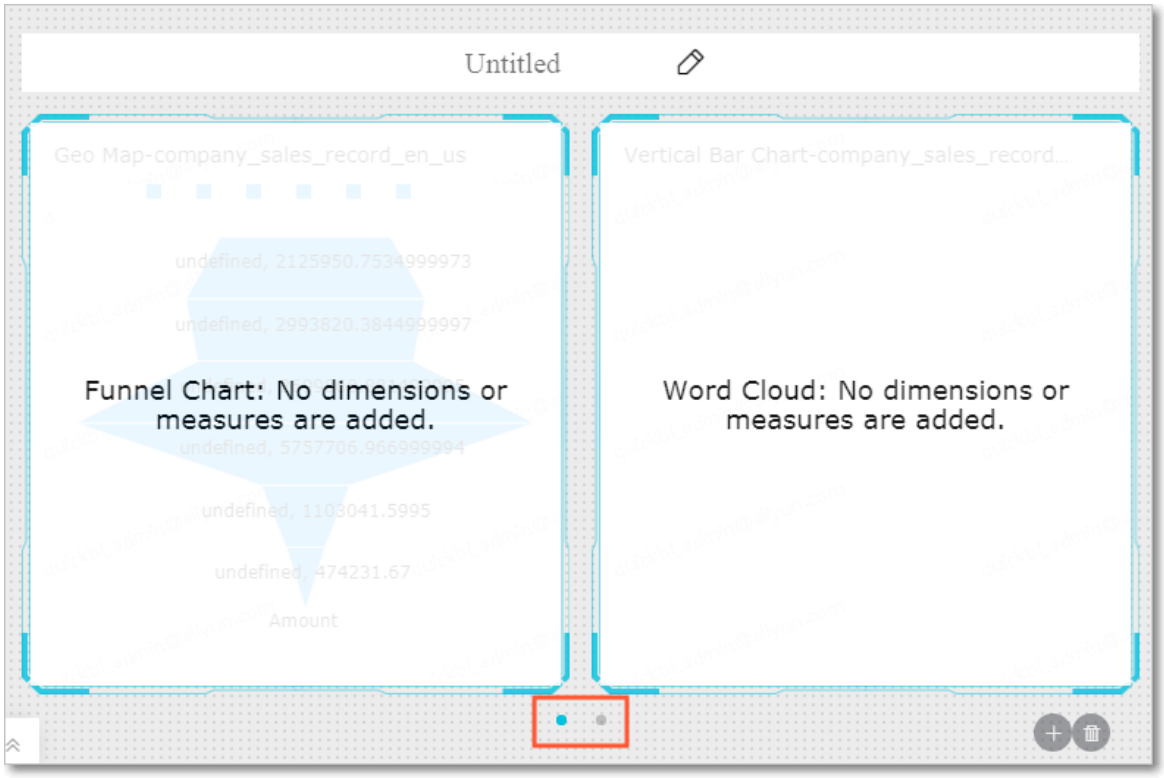
1. Click the plus sign in the lower-right corner.



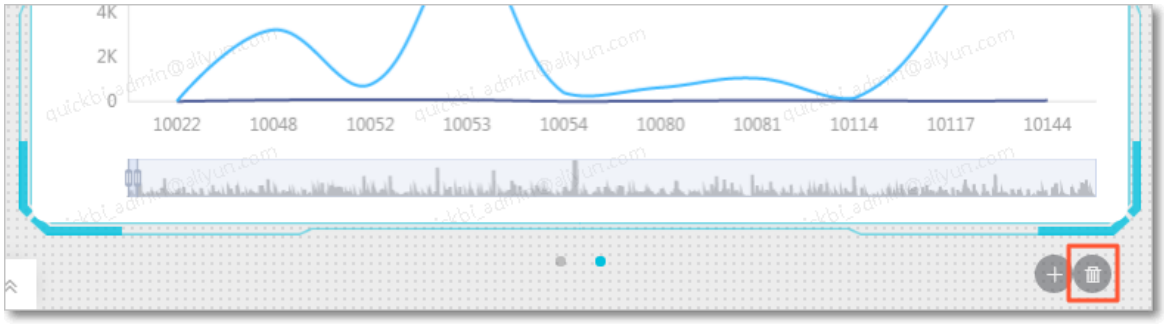
2. Add a chart to the subscreen.



3. Click the **Switch Screen** icon to switch from the current screen to another.



4. Click the **Delete** icon in the lower-right corner to delete a subscreen.



View data, export and view SQL statements, and delete a chart

1. In the target chart, click the More icon in the upper-right corner.
2. Select **View Data** to view chart data.
3. Select **Export** to export the chart data to a local device.
4. Select **View SQL Statements** to view the SQL statements.
5. Select **Delete** to delete the chart.

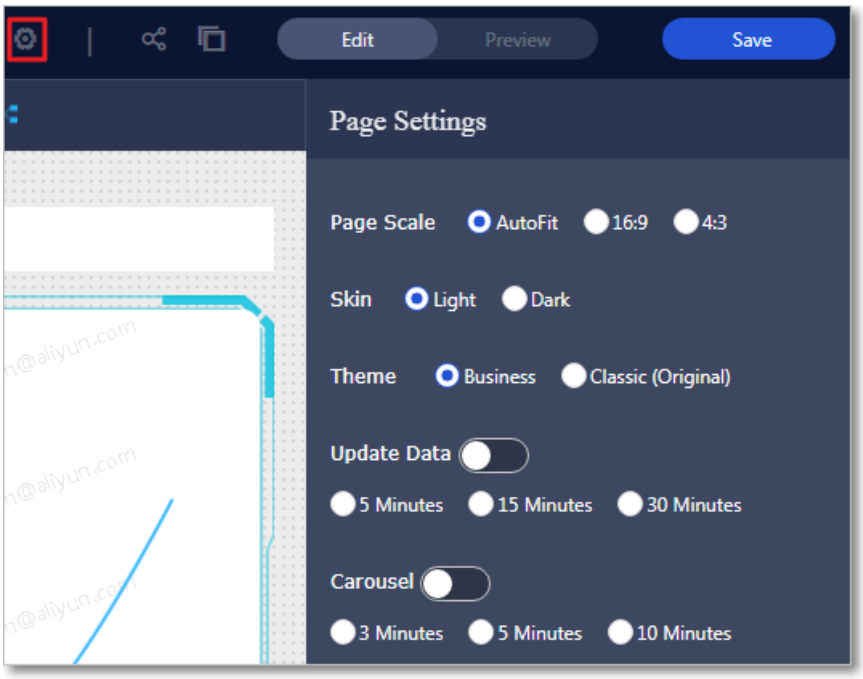
Change chart types

1. Select a chart in the display area of a dashboard.
2. In the Graphic Design area, click **Change Chart Type**.
3. Click the chart type you want.

> **Note**
>
> If the chart type fails to change, the fields of the current chart do not match those of the target chart. You must manually adjust fields before you change the chart type. The system provides instructions to help you adjust fields based on the current and target chart types. To change the chart type, you need to follow the instructions to adjust the dimensions and measures.

Page settings

You can click the **Page settings** icon to set the page scale, skin color, theme, data update interval, and time interval of data carousel.



7.4.6. Search for a dashboard

This topic describes how to search for a specific dashboard.

Prerequisites

- The Quick BI service is purchased.
- One or more dashboards are created.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Dashboards**.
3. Enter a keyword in the search box to search for the target dashboard.

7.4.7. Create a dashboard folder

This topic describes how to create a dashboard folder.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Dashboards**.
3. On the Dashboards page, click **Create Folder** in the upper-right corner.
4. In the **Create Folder** dialog box that appears, specify a name for the folder and click **OK**.

7.4.8. Rename a dashboard folder

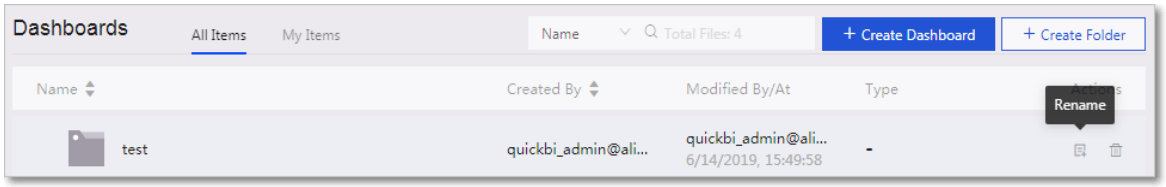
This topic describes how to rename a dashboard folder.

Prerequisites

- The Quick BI service is purchased.
- A dashboard folder is created. For information about how to create a dashboard folder, see Create a dashboard folder.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Dashboards**.
3. On the Dashboards page, find the dashboard folder you want to rename.
4. Click the **Rename** icon in the Actions column.



5. In the dialog box that appears, enter a new folder name and click **OK**.

7.4.9. Share a dashboard

This topic describes how to share a dashboard with other users.

Prerequisites

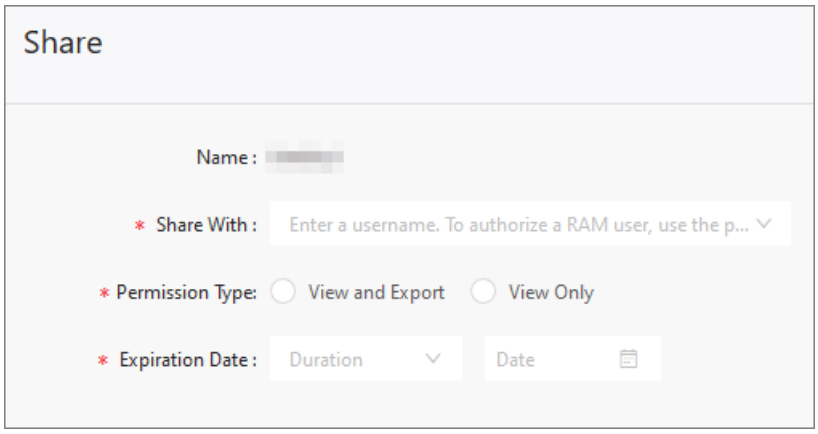
- The Quick BI service is purchased.
- A dashboard is created.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Dashboards**.
3. On the Dashboards page that appears, right-click the dashboard you want to share with others.

4. Select **Share**.
5. Specify Scope, Permission Type, and Expiration Date in the side pane that appears, as shown in Share a dashboard.

   Share a dashboard

   > **Note**
   > - In the Basic Settings section of the right-side Page Settings pane, select **Allow Download**. If you want to download a report, you must set Permission Type to **View and export**.
   > - You can set Scope to All Users, User Groups, or Users.
   > - Users can be authorized in the three scopes at the same time. Users who are authorized in one of these scopes are granted the corresponding permissions.

6. Click **Save**.

7.4.10. Make a dashboard public

After you make a dashboard public, users can access the dashboard by using a shared link.

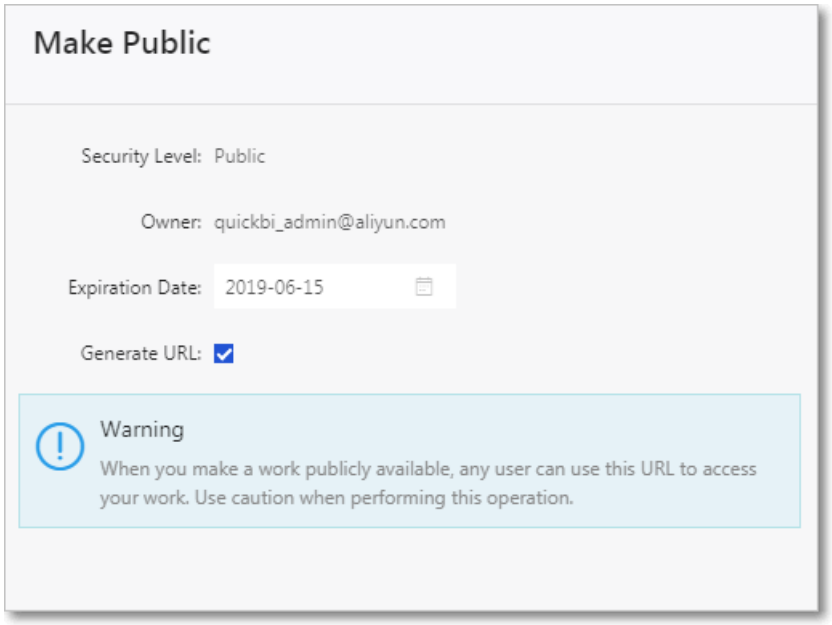
Prerequisites

- The Quick BI service is purchased.
- A dashboard folder is created. For information about how to create a dashboard folder, see Create a dashboard folder.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Dashboards**.
3. On the Dashboards page, right-click the dashboard you want to make public or click the More icon in the Actions column corresponding to the dashboard you want to make public.
4. Select **Make Public**.
5. Set an expiration date and click **Make Public**, as shown in Make a dashboard public.

   Make a dashboard public



7.5. Workbooks

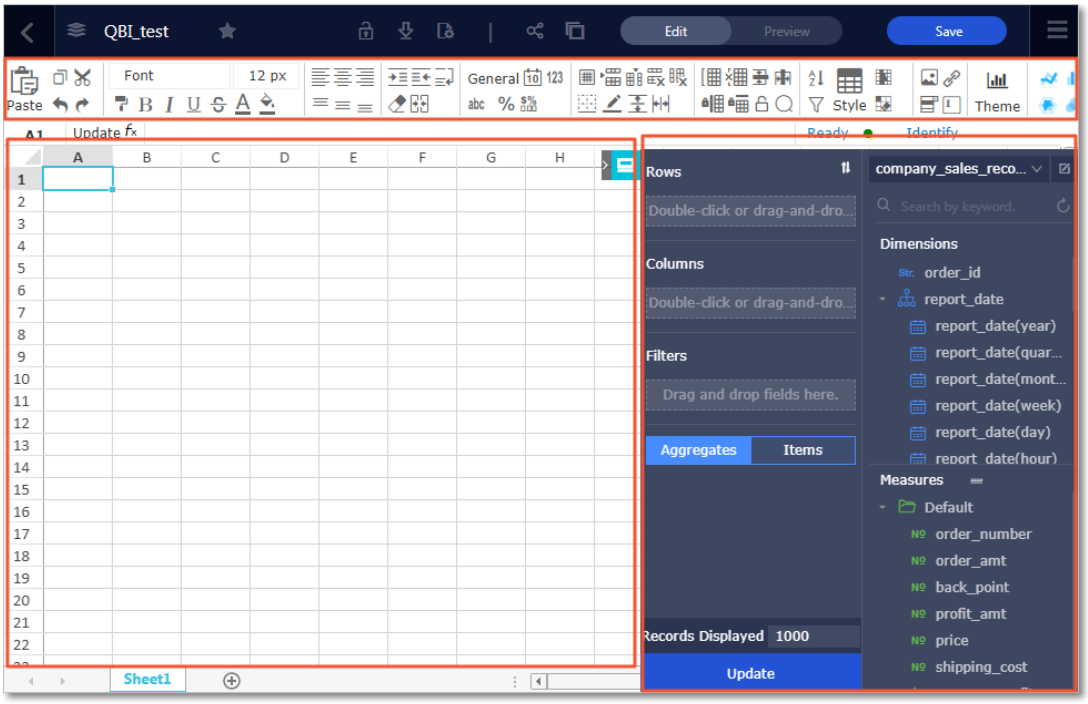
7.5.1. Overview

On the workbook edit page, you can filter and query data in a dataset. You can also visualize data by using different types of charts.

The workbook edit page contains three areas, as shown in Workbook edit page.

- Dataset selection area
- Workbook configuration area
- Workbook display area

Workbook edit page



- Dataset selection area: In this area, you can switch the current dataset to another. The fields of each dataset are displayed in the Dimensions and Measures lists based on the data types preset in the system. You can select dimensions and measures based on the data required by the chart.
- Workbook configuration area: In this area, you can select a chart type, and set the color, font, and data format of cells as needed.
- Workbook display area: In this area, you can reprocess data based on the displayed data in cells and reference data.

7.5.2. Create a workbook

This topic describes how to create a workbook.

Prerequisites

The Quick BI service is purchased.

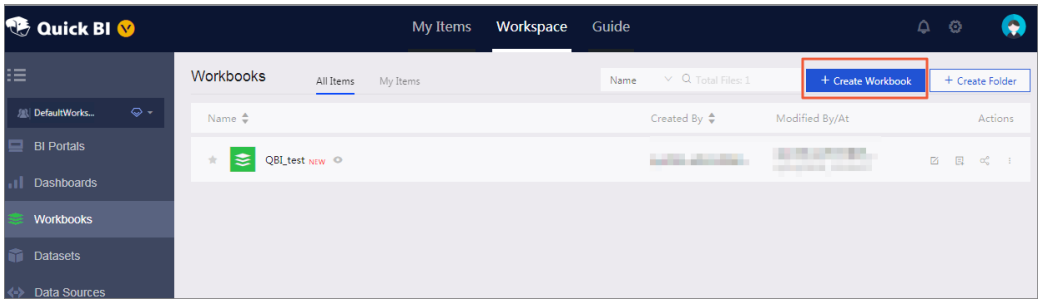
Context

You can create workbooks only in workspaces. Personal workspaces do not support workbooks.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks**.
3. On the **Workbooks** page, click **Create Workbook** to go to the workbook edit page, as shown in Create a workbook.

   Create a workbook



4. Click **Save**. In the Save Workbook dialog box that appears, enter a name for the workbook and set the location where you want to store the workbook, and click **OK**.

7.5.3. Switch to another dataset

This topic describes how to switch from the current dataset to another dataset.

Prerequisites

The Quick BI service is purchased.

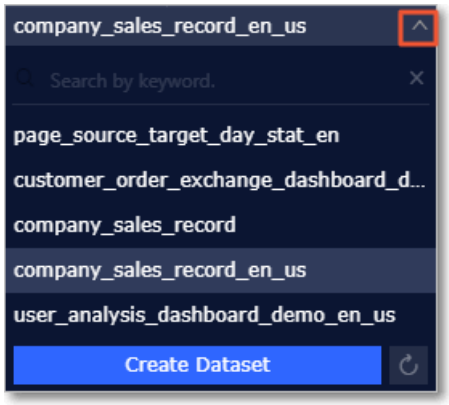
Context

If you cannot find the required dataset, go back to the dataset management page and ensure that the dataset has been created. For information about how to create a dataset, see Create datasets.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Click the workbook you want to edit.
4. On the workbook edit page, click the drop-down arrow. In the drop-down list, select or search for the dataset you want to switch to, as shown in Switch to another dataset.

Switch to another dataset



7.5.4. Search for a dimension or measure

This topic describes how to search for a specific dimension or measure.

Prerequisites

The Quick BI service is purchased.

Context

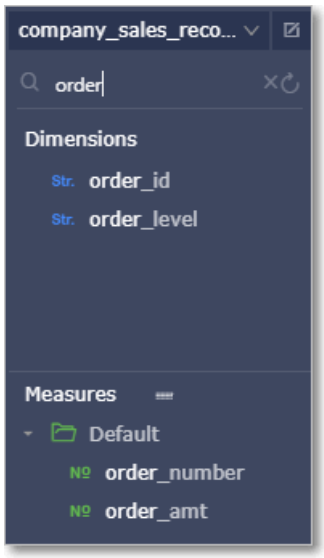
After you select a dataset, the system displays the dimensions in the Dimensions list and measures in the Measures list.

For information about how to edit dimensions and measures, see Edit a dimension and Edit a measure.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Enter a keyword of the field you want to search in the search box.
4. Click the Search icon to search for the field.

Search for a field



7.5.5. Set the font

You can set a font for a specific text, such as the font size, font color, font style, and background color.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the

Workbooks page.

3. On the Workbooks page, click the workbook that you want to set the font.

   For information about how to create a workbook, see Create a workbook.

4. Click the specific icon to set the font size and style. Set the font style
   - Click the font area.
   - In the drop-down list that appears, select the target font, as shown in Select a font.

     Select a font

   Set the font size
   - Click the font size area.
   - In the drop-down list that appears, select the target font size, as shown in Select a font size.

     Select a font size

7.5.6. Set the alignment mode

You can set the alignment mode to adjust the layout of text.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook that you want to set the alignment mode.

   For information about how to create a workbook, see Create a workbook.

4. Click the specific alignment mode icon to adjust the layout of the text, as shown in Alignment modes.

   Alignment modes

7.5.7. Set text and number formats

You can set the format to display texts and numbers in a workbook.

Prerequisites

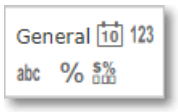
The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook for which you want to set the text and number format.

   For information about how to create a workbook, see Create a workbook.
4. Click the specific icon to set the format, as shown in Display formats.

   Display formats

Parameter	Description
General	The General format directly displays numbers the way that you type them.
Date	The Date format displays data in the YYYY-MM-DD format.
Number	The Number format aligns numbers along the right side. It rounds numbers to two decimal places. You can double-click the cell or adjust the column width to show the complete number.
String	The String format aligns strings along the left side. You can double-click the cell or adjust the column width to show the complete string.
Percentage	The Percentage format aligns numbers along the right side. It rounds numbers to two decimal places. You can double-click the cell or adjust the column width to show the complete number.

7.5.8. Set the style, cell, and pane

You can change the style, cell, and pane settings to adjust the gridlines, row heights, and border styles.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.

2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook for which you want to change the style, cell, or pane settings.

   For information about how to create a workbook, see Create a workbook.
4. On the workbook edit page, click the specific icons to adjust the layout of the workbook, as shown in Style, cell, and pane settings.

   Style, cell, and pane settings

Parameter	Description
Gridlines	The gridlines are displayed by default. You can click the **Gridlines** icon to hide gridlines.
Borders	You can click the Borders icon to display a top border, bottom border, left border, right border, outside borders, or all borders, or remove all borders.
Border Color	You can specify a color for the borders.
Insert and Delete	You can insert rows and columns into a workbook, or delete rows and columns from a workbook. You can also insert and delete a workbook.
AutoFit Column Width	Double-click the AutoFit Column Width icon. The column width is automatically adjusted.
AutoFit Row Height	Double-click the AutoFit Row Height icon. The row height is automatically adjusted.

7.5.9. Insert images, hyperlinks, and drop-down list boxes

This topic describes how to insert images, hyperlinks, and drop-down list boxes into a workbook.

Prerequisites

The Quick BI service is purchased.

Procedure

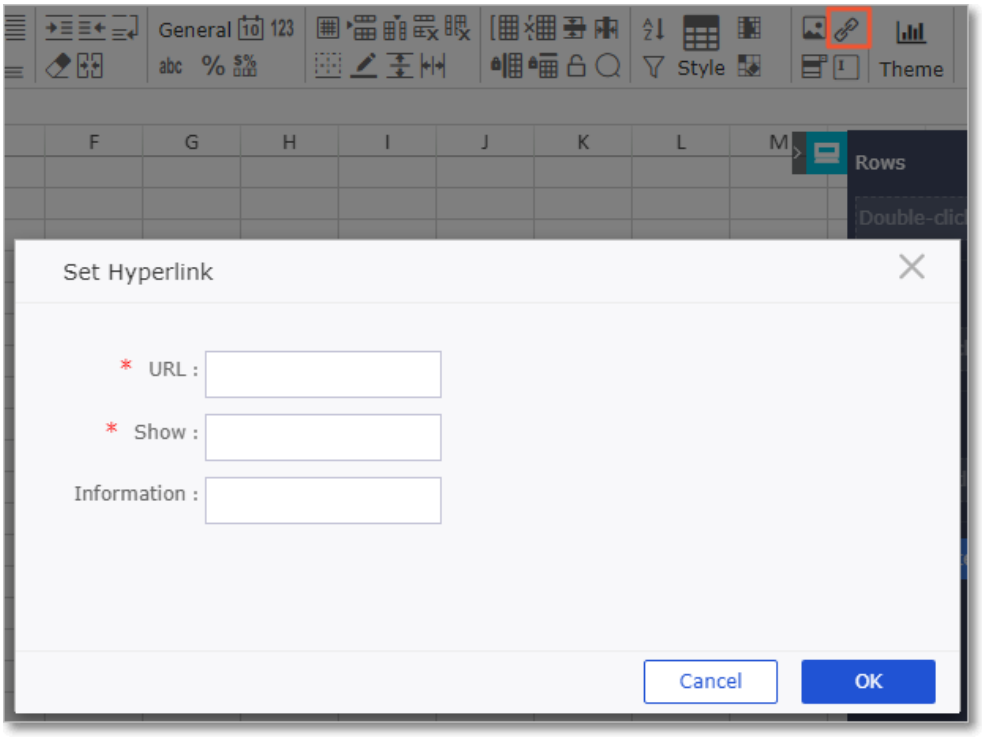
1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook for which you want to insert images, hyperlinks,

or drop-down list boxes.

For information about how to create a workbook, see Create a workbook.

- Insert an image
  - Click the **Upload Image** icon.
  - In the Upload Image dialog box that appears, click **Select File** and select an image to upload.
  - Click **OK** to insert the image.
- Insert a hyperlink
  - Click the **Hyperlink** icon.
  - In the Set Hyperlink dialog box that appears, enter the hyperlink that you want to insert and the text to display, as shown in Insert a hyperlink.

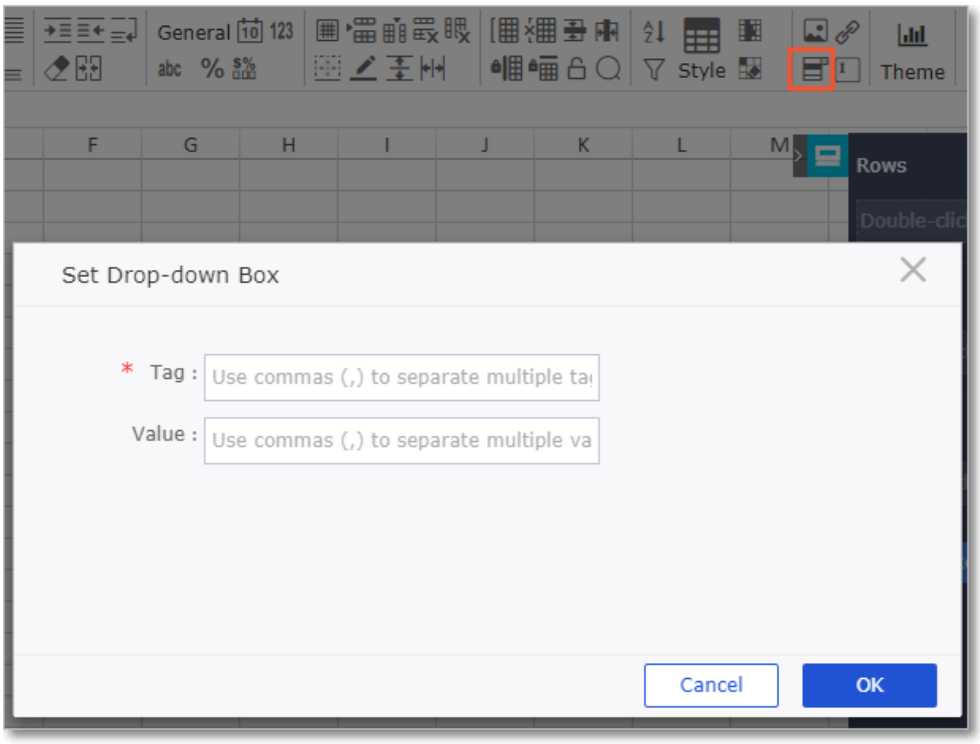
    Insert a hyperlink



  - Click **OK** to insert the hyperlink.
- Insert a drop-down list box
  - Click the **Drop Down** icon.

- In the Set Drop-down Box dialog box that appears, specify tags and values, as shown in Set a drop-down box.

  Set a drop-down box



- Click **OK** to insert the drop-down list box.

7.5.10. Set the workbook style

This topic describes how to set the style for a workbook.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook for which you want to set the style.

   For information about how to create a workbook, see Create a workbook.
4. Click the **Style** icon.
5. Select an appropriate table style.

7.5.11. Set conditional formatting

This topic describes how to set conditional formatting, for example, highlight specific numbers or add an upward or downward arrow to indicate an ascending or descending order in a workbook.

Prerequisites

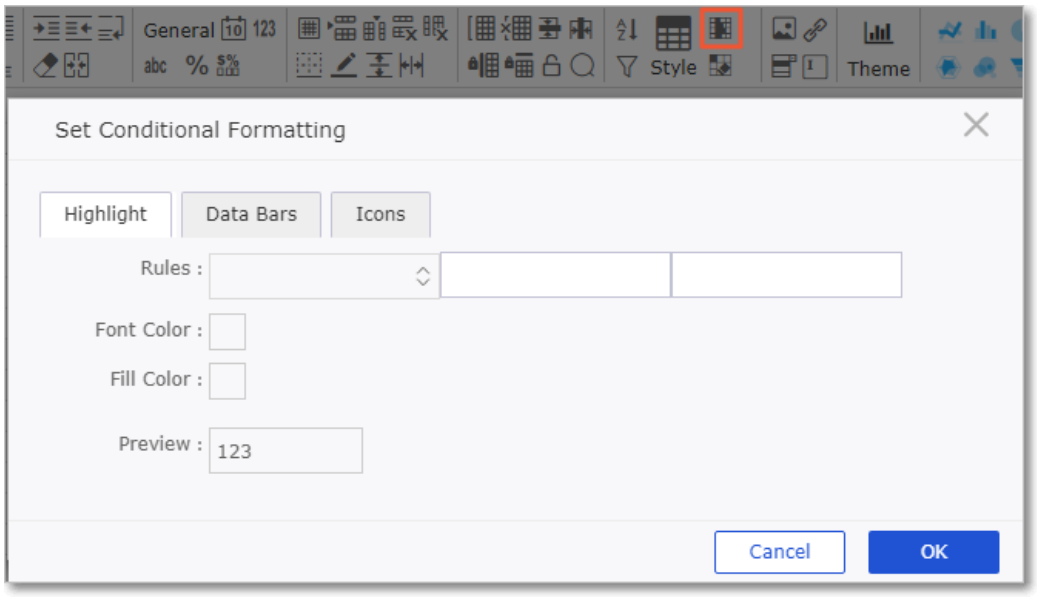
The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. On the Workbooks page, click the workbook for which you want to set conditional formatting.

   For information about how to create a workbook, see Create a workbook.
4. Click the **Set Conditional Formatting** icon.
5. In the Set Conditional Formatting dialog box that appears, click the **Highlight** tab, as shown in Highlight settings.

   Highlight settings

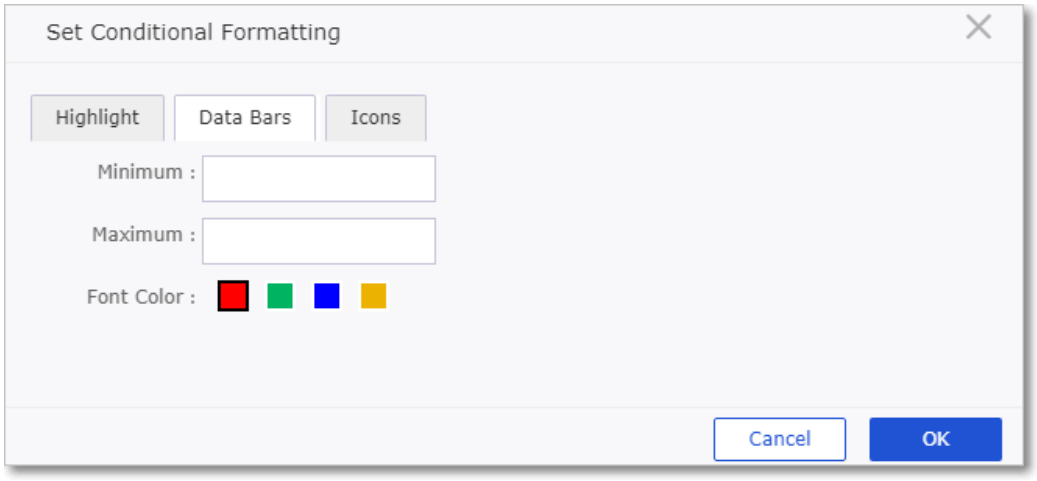


Parameter	Description
Rules	Click the drop-down icon to select a highlighting rule from the drop-down list, and specify a value or value range in the input boxes.
Font Color	Click the Color icon and select a color.
Fill Color	Click the Color icon and select a color.

Parameter	Description
Preview	Displays the highlight effect after you set the colors.

6. Click the **Data Bars** tab, as shown in Data Bars.

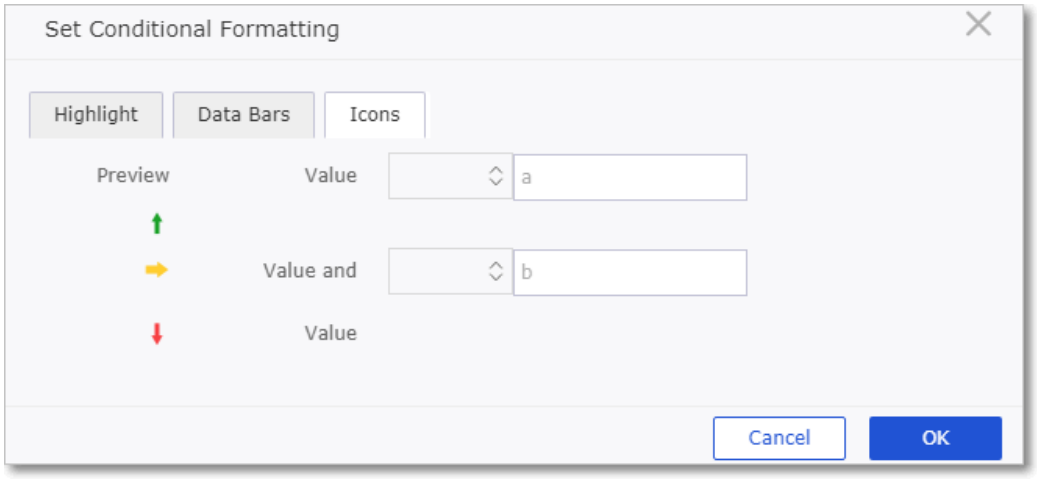
Data Bars



Parameter	Description
Minimum	Enter a value in the input box.
Maximum	Enter a value in the input box.
Font Color	Click the Color icon and select a color.

7. Click the **Icons** tab, as shown in Icons.

Icons



Click the drop-down icon, select a mathematical notation from the drop-down list, and enter a value in the input box. A green, yellow, or red arrow appears next to the values that fit the specific value ranges.

8. After you set the parameters, click **OK**.

7.5.12. Search for a workbook

This topic describes how to search for a specific workbook.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Enter a keyword in the search box.
4. Click the Search icon to search for the workbook.

7.5.13. Create a workbook folder

This topic describes how to create a workbook folder. Workbook folders help you manage workbooks.

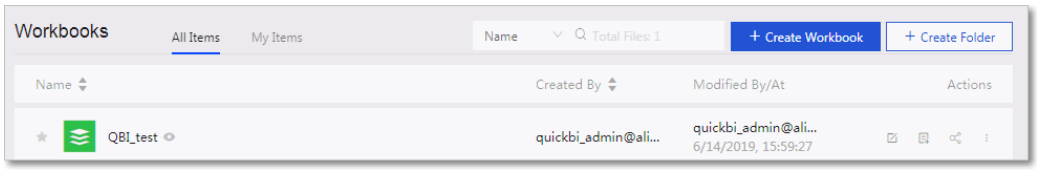
Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Click **Create Folder**, as shown in Create a folder.

   Create a folder



4. In the Create Folder dialog box that appears, specify a name for the folder and click **OK**.

7.5.14. Rename a workbook folder

This topic describes how to rename a workbook folder.

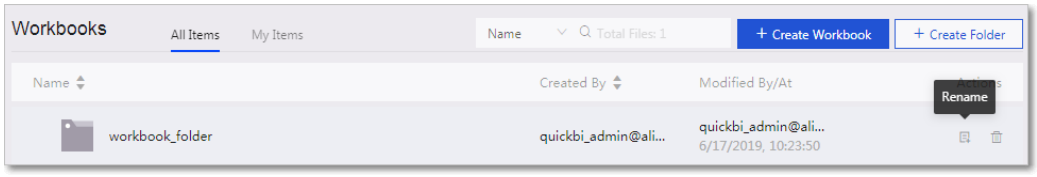
Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Find the workbook folder you want to rename and click the **Rename** icon in the Actions column, as shown in Rename a workbook folder.

   Rename a workbook folder



4. In the **Rename** dialog box that appears, enter a new folder name and click **OK**.

7.5.15. Share a workbook

This topic describes how to share a workbook with other users.

Prerequisites

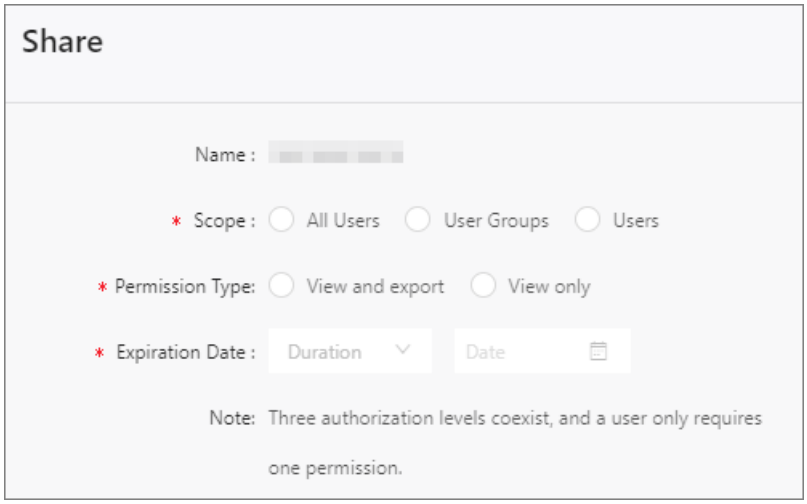
- The Quick BI service is purchased.
- A workbook is created.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Find the workbook that you want to share with other users.
4. Click the **Share** icon in the Actions column, as shown in Share a workbook.

   Share a workbook

5. Specify Scope, Permission Type, and Expiration Date in the side pane that appears.



> **Note**
> - You can set Scope to All Users, User Groups, or Users.
> - Users can be authorized in the three scopes at the same time. Users who are authorized in one of these scopes are granted the corresponding permissions.

6. Click **Save**.

7.5.16. Make a workbook public

After you create a workbook, you can make it public to allow other users to access the workbook.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** page, click **Workbooks** to go to the **Workbooks** page.
3. Find the workbook that you want to make public and click the **More** icon in the Actions column.
4. Select **Make Public**.
5. In the Make Public pane that appears, specify an expiration date.
6. Select **Generate URL** and click **Make Public**.

7.6. BI portals

7.6.1. Overview

A BI portal is a collection of dashboards, workbooks, and external links organized with menus. You can create a BI portal to perform complex thematic analysis with navigation panes.

7.6.2. Create a BI portal

This topic describes how to create a BI portal.

Prerequisites

The Quick BI service is purchased.

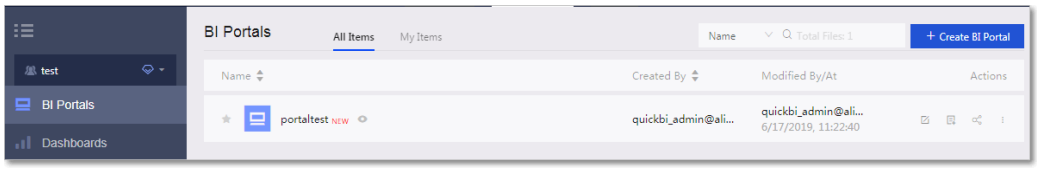
Context

You can create BI portals in workspaces only. Personal workspaces do not support BI portals.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **BI Portals**.
3. On the BI Portals page, click **Create BI Portal** in the upper-right corner, as shown in Create a BI portal.

   Create a BI portal

4. On the Page Settings page, set the parameters and click **Save** in the top navigation bar.

7.6.3. Page settings

This topic describes how to edit a BI portal page, such as the title, layout, logo, and footer.

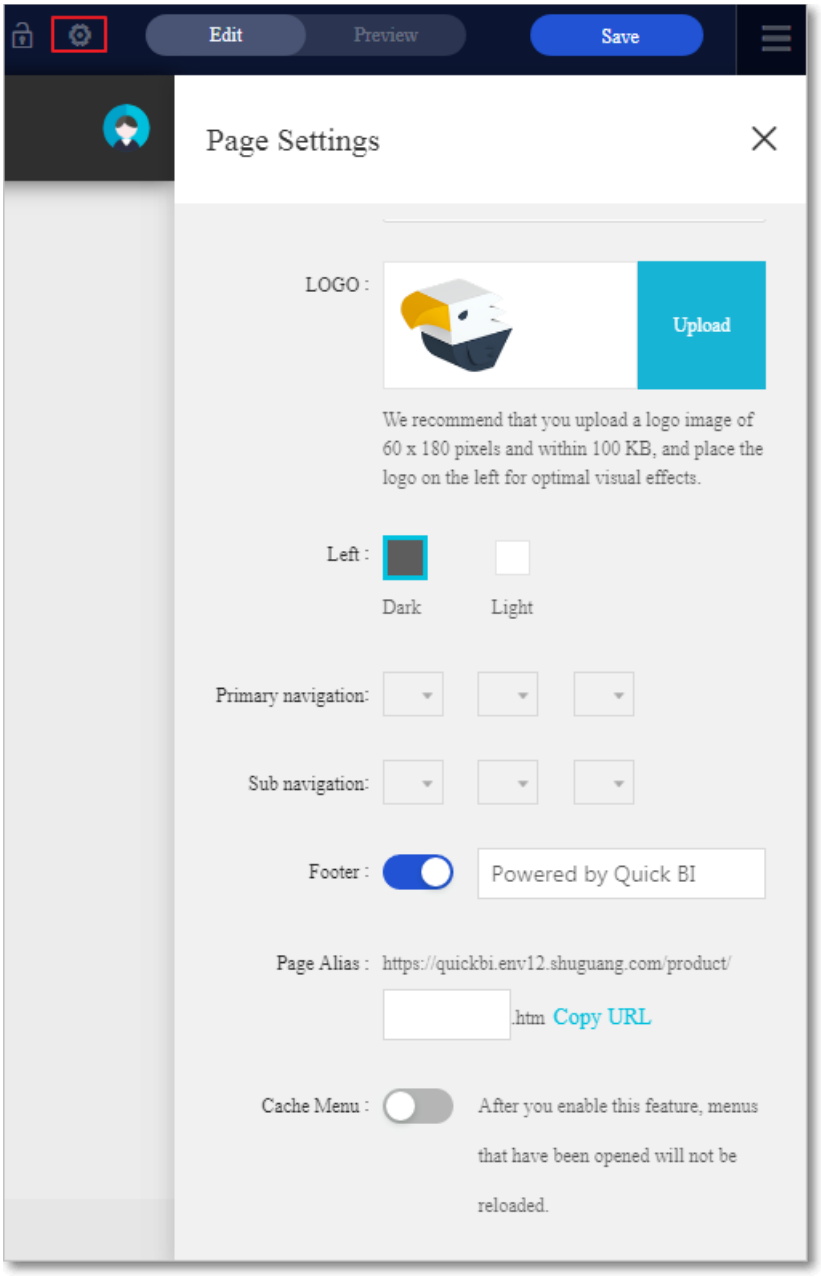
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **BI Portals**.
3. On the BI Portals page, click a BI portal.

   For information about how to create a BI portal, see Create a BI portal.

4. Click the **Settings** icon in the top navigation bar to edit the BI portal page, as shown in Template settings.

   Template settings



5. Click **Save**.

7.6.4. Menu settings

This topic describes how to edit the menu content, such as menu titles and URLs, on the Menu Settings tab.

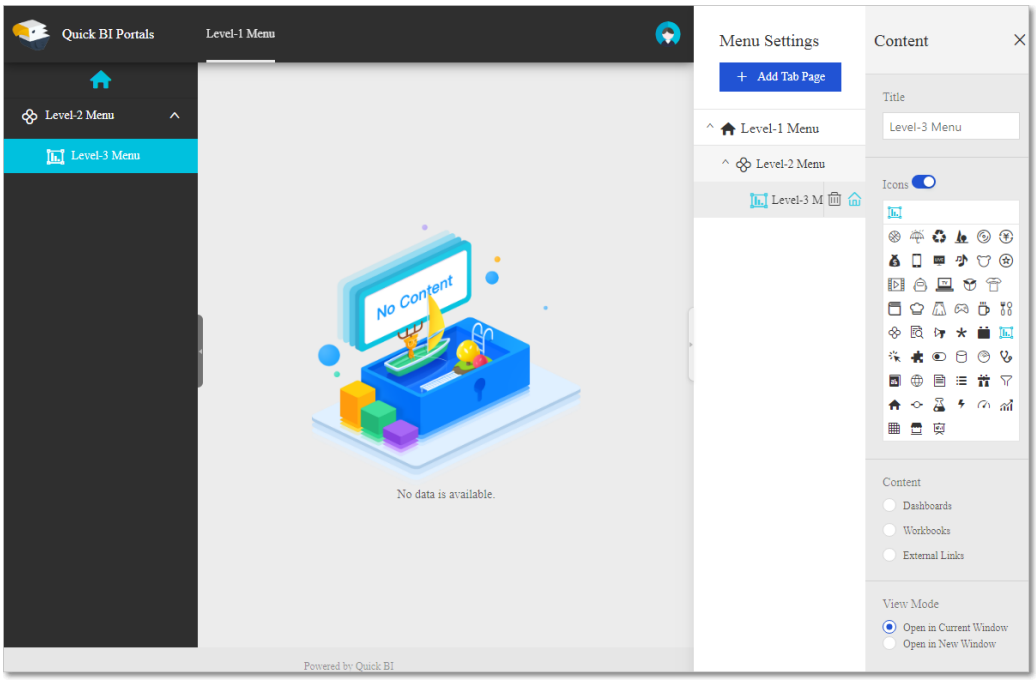
Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the **Workspace** tab, click **BI Portals**.
3. On the BI Portals page, click a BI portal.

   For information about how to create a BI portal, see Create a BI portal.

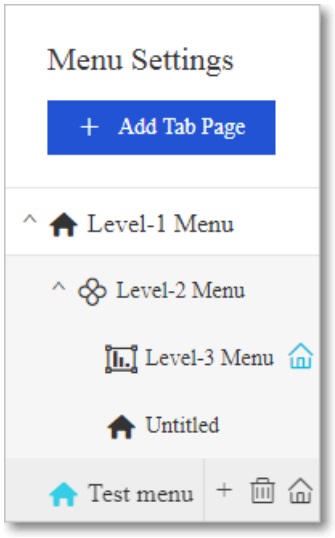
4. In the left-side navigation pane, click the target menu, and edit the menu on the right-side **Content** tab, as shown in the following figure.

   Menu settings

   - On the **Menu Settings** tab, you can edit menu settings, as shown in the following figure.

     Edit the menu structure

   - You can add a dashboard or workbook as a menu.

5. Click **Save**.

7.7. Organization

7.7.1. Overview

An organization typically refers to a small or medium enterprise, public institution, college department, or a department of a large enterprise.

If your organization has a large number of members, requires multiple members to collaborate on data analysis, and has high requirements on data security, Quick BI provides the following features to meet your requirements:

- Different departments have access to different reports.
- Members with different roles have access to different data.

Members in an organization are classified into two types: administrators and common members.

7.7.2. Create an organization

This topic describes how to create an organization.

Prerequisites

Before you create an organization, you must create an Apsara Stack tenant account in the Apsara Stack console. Each Apsara Stack tenant account can be used to create or join in only one organization. Ensure that your Apsara Stack tenant account has neither been used to create an organization nor been added to an organization before.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI** console, or in a workspace, click the **Settings** icon.

3. Click **Organization** in the left-side navigation pane.
4. Select the **Agree** check box and click **Create Organization**.
5. In the **Create Organization** dialog box, enter an organization name.

7.7.3. Modify organization information

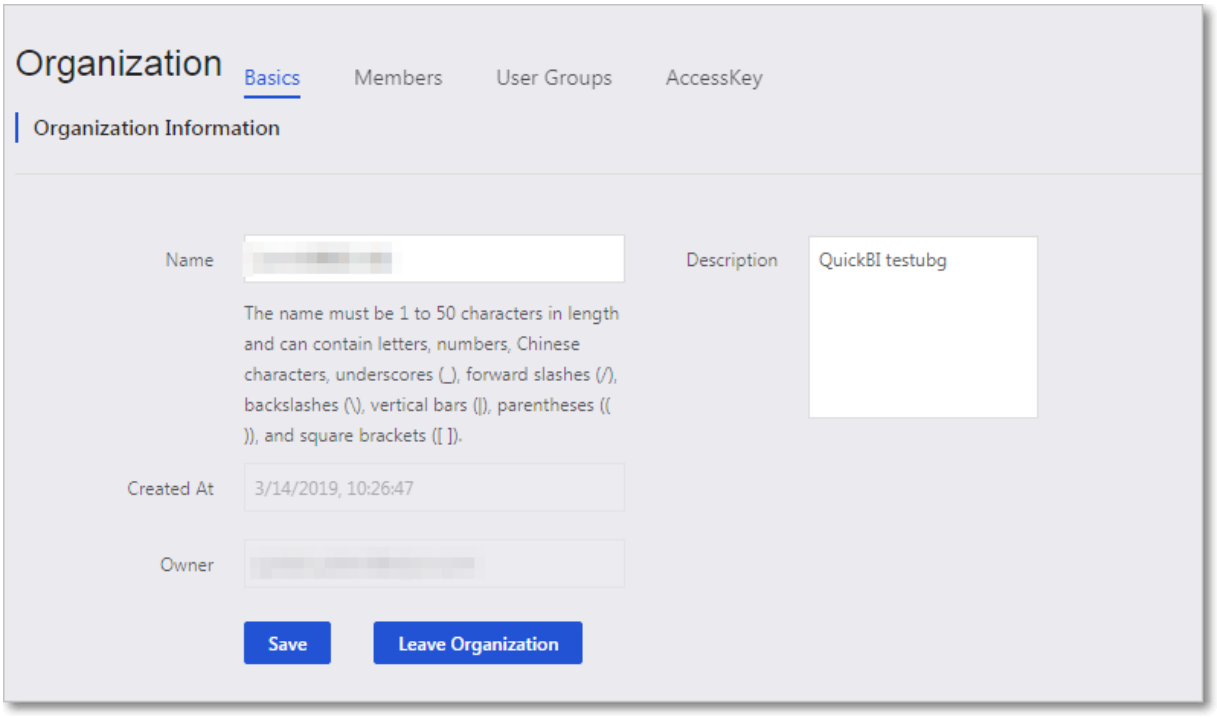
Quick BI allows you to modify the information about an organization as needed.

Prerequisites

The Quick BI service is purchased.

Context

Administrators of an organization have the permissions to modify the information about the organization.



Administrators of an organization have the permissions to add members to the organization to collaborate on tasks.

Administrators of workspaces have the permissions to add members to their workspaces based on the roles and responsibilities of the members. You can create workspaces that correspond to the departments of the organization. For example, if an organization has a sales department and an HR department, administrators can create a workspace for each of the two departments, and then add the employees as members to the corresponding workspaces.

Only administrators of an organization have permissions to manage members in the organization. By default, the creator of an organization is one of the administrators of the organization.

Members in an organization are classified into administrators and common members.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console**, or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Basics** tab.
4. Modify the organization information and click **Save**.

7.7.4. Leave an organization

Quick BI allows you to leave an organization.

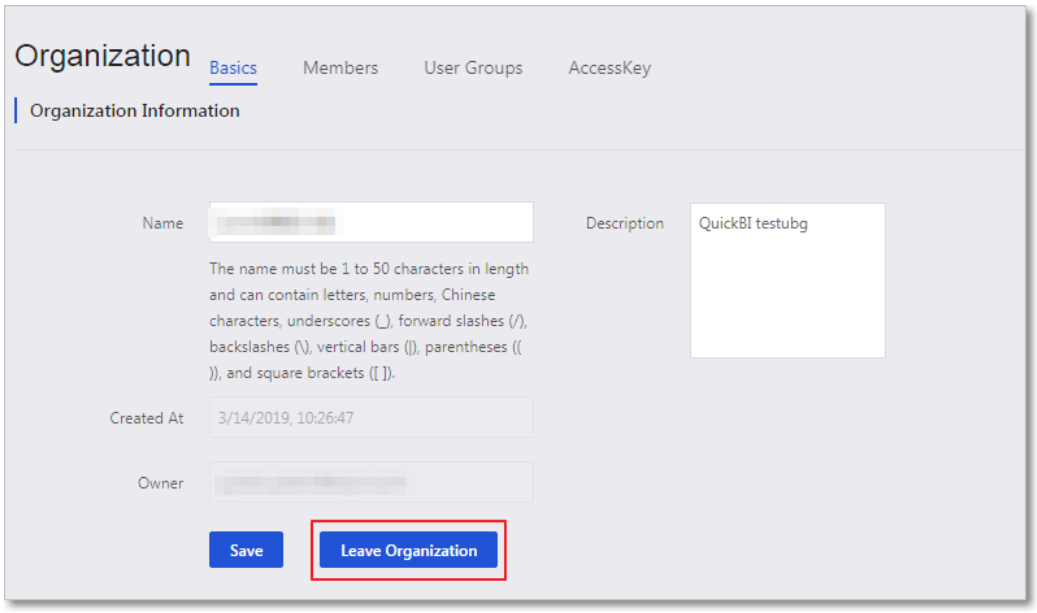
Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console**, or in a workspace, click the **Settings** icon.
3. On the Organization page, click the Basics tab and then click **Leave Organization**, as shown in Leave an organization.

Leave an organization



7.7.5. Add a member to an organization

You can add a member to an organization by **adding an Apsara Stack tenant account** or **adding a RAM user**.

Prerequisites

- The Quick BI service is purchased.
- The Apsara Stack tenant account is obtained.

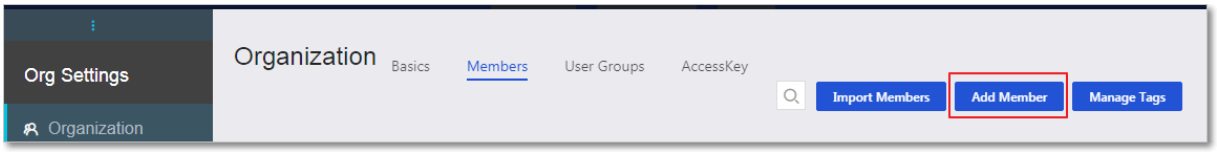
  Each department created in department management in the Apsara Stack console has an Apsara Stack tenant account. The Apsara Stack tenant account of a user can be queried by using the department to which the user belongs.

- The RAM user information is obtained.

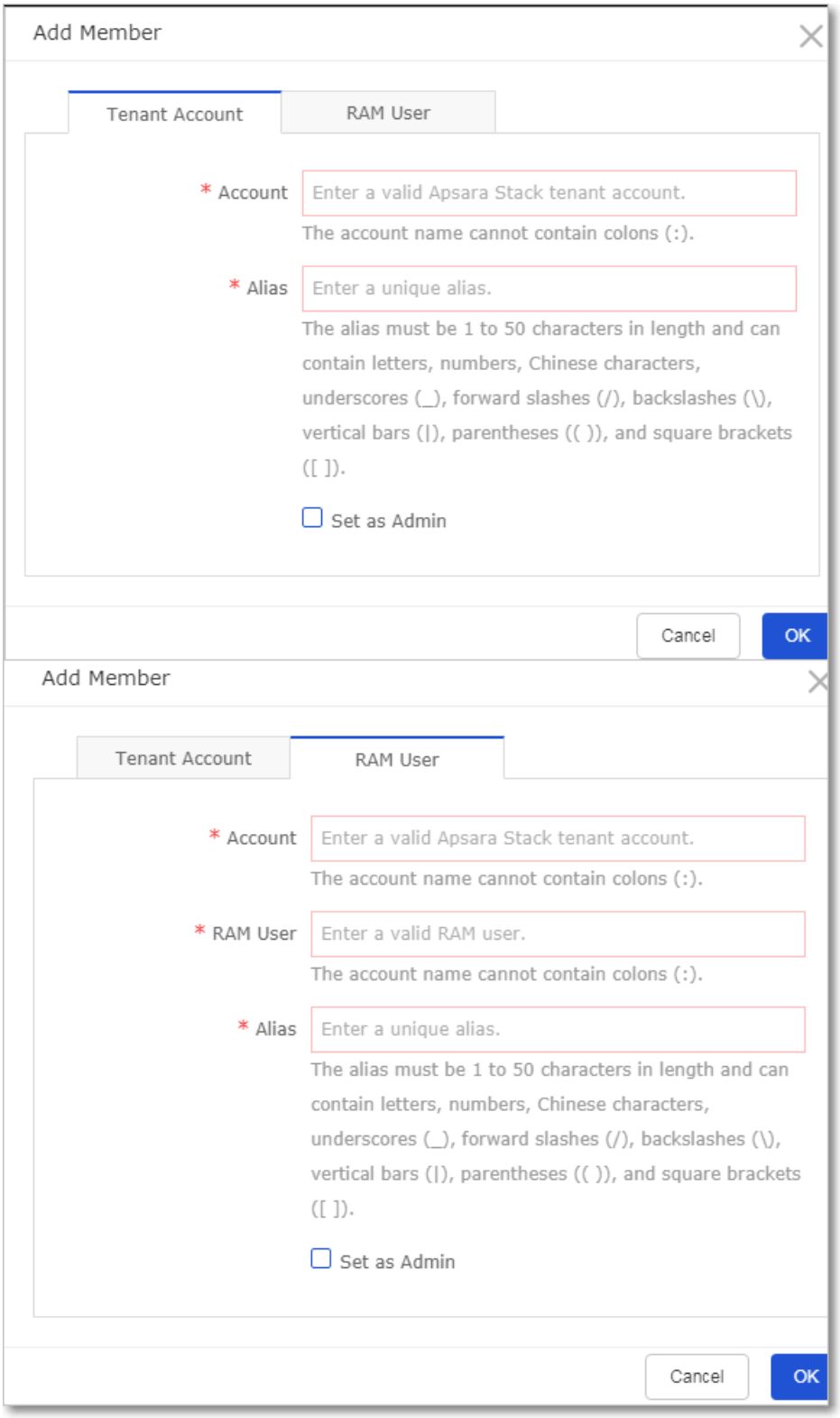
  When you **add a RAM user** in the Quick BI console, you must know both the Apsara Stack tenant account and the RAM user information. The RAM user name is displayed in user management.

Context

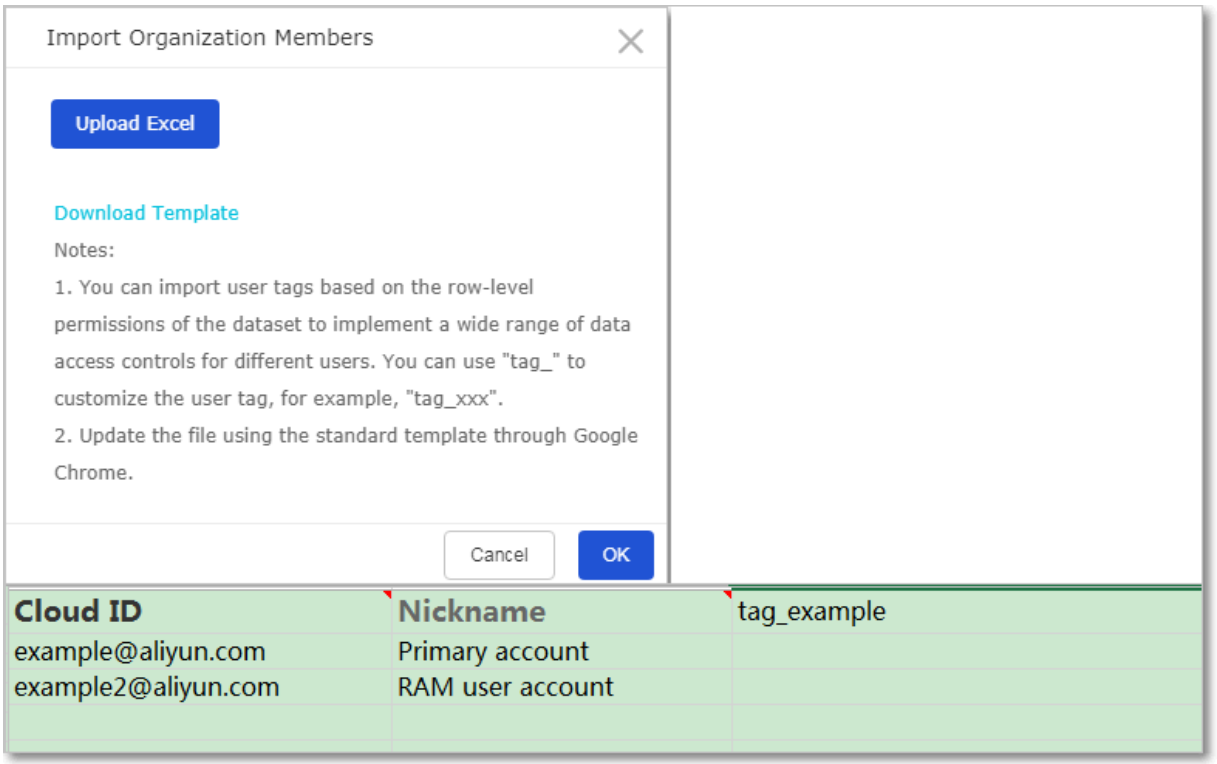
Quick BI allows you to add one member at a time or multiple members at the same time to an organization.



When you add one member at a time, you can add an Apsara Stack tenant account or a RAM user.


Add Member
Tenant Account
RAM User
* Account
Enter a valid Apsara Stack tenant account.
The account name cannot contain colons (:).
* Alias
Enter a unique alias.
The alias must be 1 to 50 characters in length and can contain letters, numbers, Chinese characters, underscores (_), forward slashes (/), backslashes (\), vertical bars (|), parentheses (( )), and square brackets ([ ]).
Set as Admin
Cancel
OK
Add Member
Tenant Account
RAM User
* Account
Enter a valid Apsara Stack tenant account.
The account name cannot contain colons (:).
* RAM User
Enter a valid RAM user.
The account name cannot contain colons (:).
* Alias
Enter a unique alias.
The alias must be 1 to 50 characters in length and can contain letters, numbers, Chinese characters, underscores (_), forward slashes (/), backslashes (\), vertical bars (|), parentheses (( )), and square brackets ([ ]).
Set as Admin
Cancel
OK

To add multiple members at the same time, you need to download a template and enter the Apsara Stack tenant accounts and aliases of the members you want to add into the template. Pay attention to the differences between the Apsara Stack tenant account and the RAM user information. When you add a member by using RAM user information, the format of the user account is Apsara Stack tenant account:RAM user.



Add a member to an organization

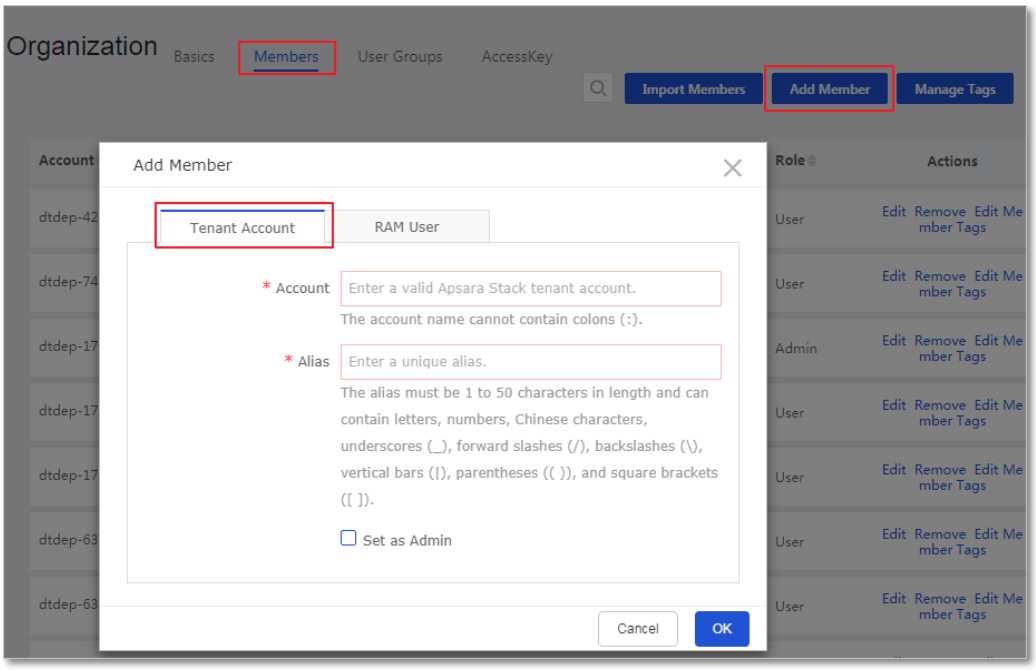
Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI** console, or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. Click **Add Member**.

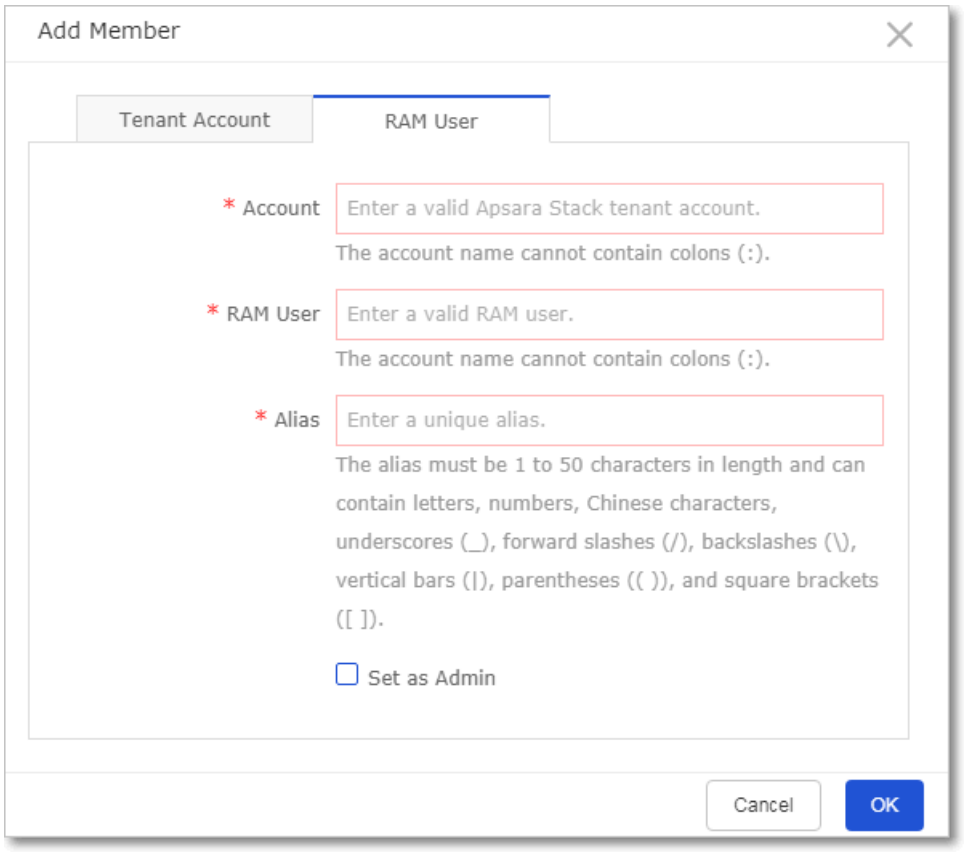
Operation	Procedure
Add an Apsara Stack tenant account	i. In the Add Member dialog box that appears, click the **Tenant Account** tab. ii. Enter the **Apsara Stack tenant account** and **alias**, and select the Set as Admin check box as needed. iii. Click **OK** to add the member.

Operation	Procedure
Add a RAM user	i. In the Add Member dialog box that appears, select **RAM User**. ii. Enter the **Apsara Stack tenant account**, **RAM user**, and **alias**, and select the Set as Admin check box as needed. iii. Click **OK** to add the member. **Note** If the user has been added to another organization, the system prompts an error.

Add an Apsara Stack tenant account



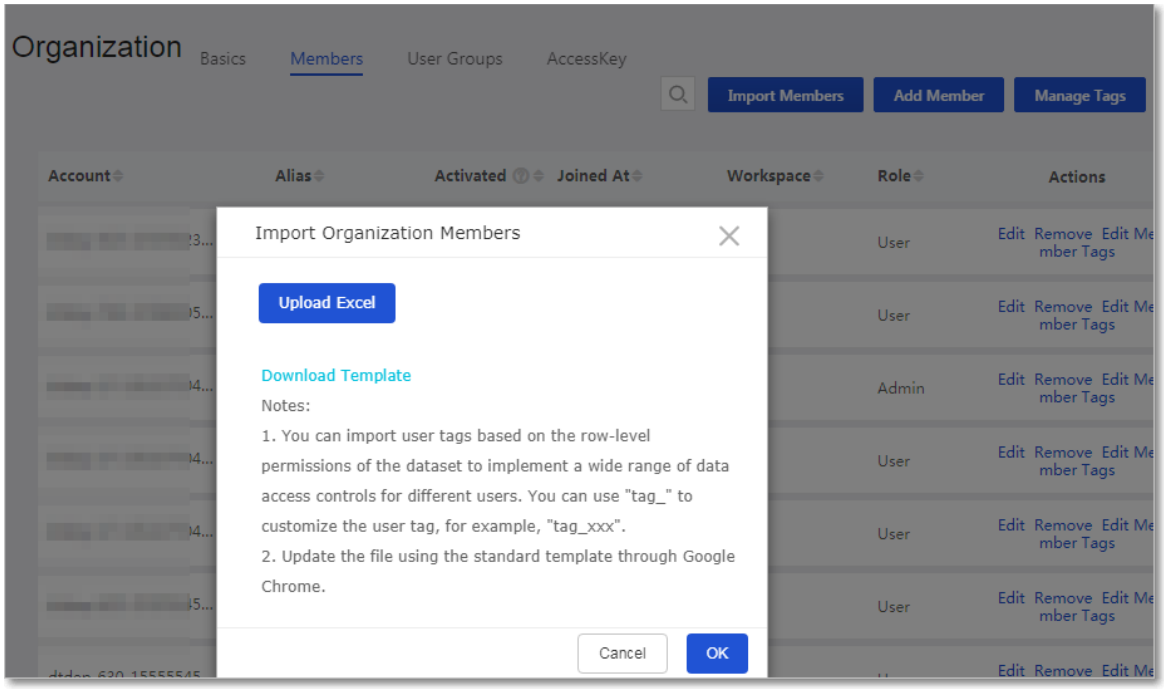
Add a RAM user



Add multiple members at a time

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI** console, or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. On the Organization page, click **Import Members**.
5. In the Import Organization Members dialog box that appears, click **Upload Excel** and select the local file that contains the member list.



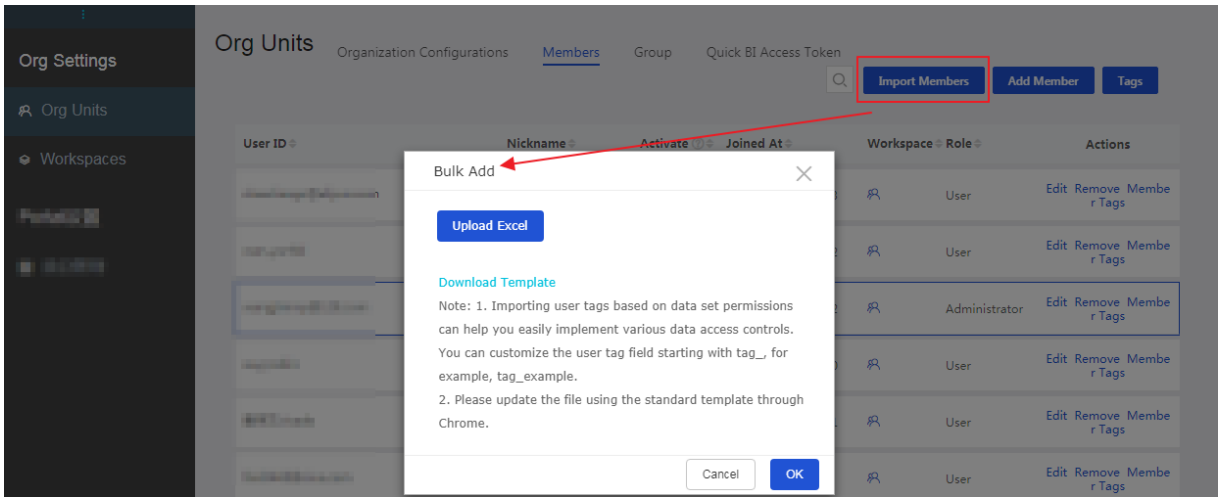
6. Click **OK** to add the members.

7.7.6. Manage member tags

This topic describes how to manage member tags, which are used to configure row-level permissions for datasets.

For information about how to configure row-level permissions, see Set row-level permissions.

You can click the **Import members** icon to add member tags.



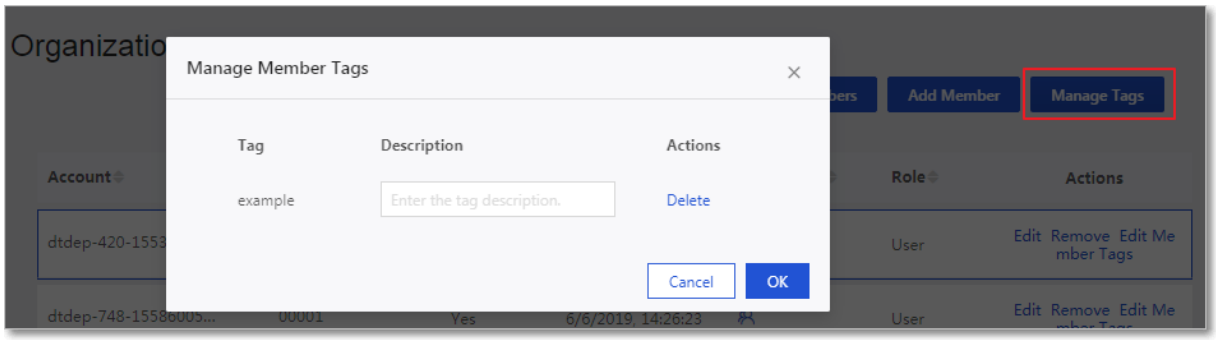
You can click **Download Template** to download the member template. The following figure shows member information.

Account	Nickname	tag_tagArea	tag_tagProvince
example1@aliyun.com	example1	East	Anhui
example2@aliyun.com	example2	East	Anhui

> **Note** If you do not need to set row-level permissions for a member, set the member tag to $ALL_MEMBERS$.

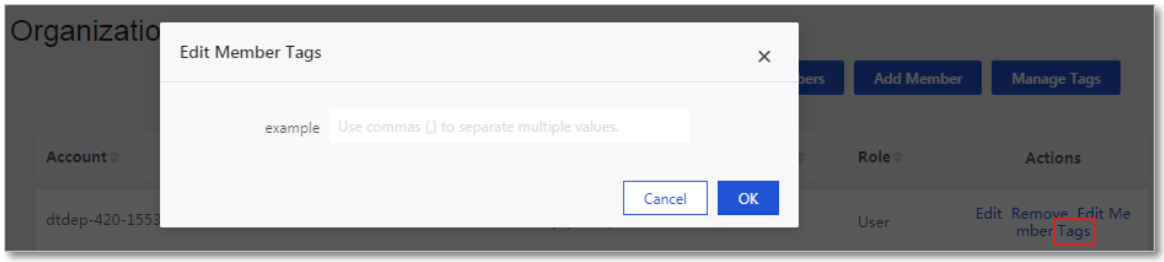
Manage member tags

You can manage tags as shown in the following figure.



Edit a tag

1. Find the member for which you want to set the row-level permissions, and click Edit Member Tags in the Actions column.
2. In the Edit Member Tags dialog box that appears, enter a new tag and click **OK**.



7.7.7. Edit a member

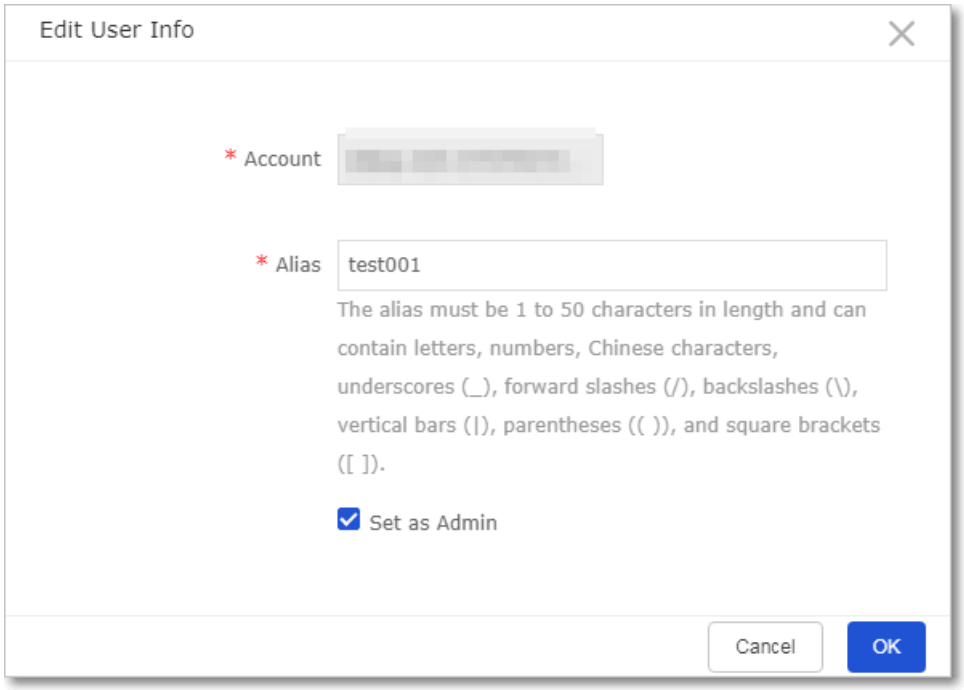
You can set an alias and role for a member in the organization. This makes it easier for you to search for members in the organization.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. Find the member you want to edit and click **Edit** in the Actions column.
5. Modify the information about the member.



6. Click **OK** to save the changes.

7.7.8. Remove a member

This topic describes how to remove a member from an organization.

Prerequisites

The Quick BI service is purchased.

Context

Only administrators of an organization have the permissions to remove members from the organization. Before you remove a member that has been added to a workspace, you must remove the member from the workspace. Otherwise, you cannot remove the member from the organization.

> **Note** The removal operation is irreversible. If a member that you previously removed is needed, you need to add the member to the organization again. Remove a member with caution.

Procedure

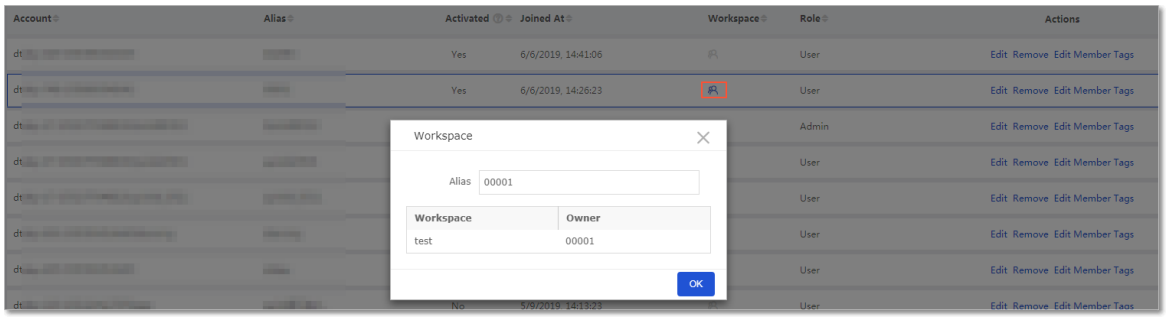
1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. Find the member you want to remove and click **Remove** in the Actions column.
5. Click **OK** to remove the member.

7.7.9. Query the workspace to which a user belongs

You can query the workspace to which a user belongs.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. Find the organization member that you want to query and click the **Workspace** icon to view the workspace that the user belongs to.

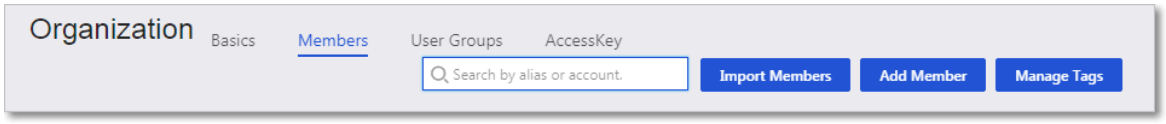


7.7.10. Search for a member of an organization

You can search for a specific member in an organization by its alias or Alibaba Cloud account.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. On the Organization page, click the **Members** tab.
4. Enter the alias or Apsara Stack tenant account of the member you want to search for in the search box, and click the **Search** icon.



7.7.11. Workspaces

7.7.11.1. Overview

A workspace is managed by its administrators. The role of a workspace administrator is assigned to members by the organization administrator that creates the workspace. A workspace administrator can specify other members in the workspace as administrators of the workspace.

A workspace administrator can:

- Create a workspace
- Modify a workspace
- Set a default workspace

7.7.11.2. What is a workspace?

A workspace enables members in the same organization to collaborate on tasks. In a workspace, each member is assigned a role to perform operations such as creating and modifying data sources, datasets, workbooks, dashboards, and BI portals. Data objects are stored in their workspaces. Each workspace has its own data objects.

A workspace has the following properties:

- Name
- Description
- Function permissions: Data objects can be shared and made public by default.
- Preference settings:
  - Use physical field names as dimension and measurement names.
  - Use field annotations as dimension and measurement names.

  Datasets created in this workspace are named based on the new settings. Existing datasets are not affected.

Members of a workspace can be assigned the same role or different roles. The roles in a workspace include:

- Administrator
- Developer
- Analyst
- Viewer

Mapping between roles and permissions

Permissions of a role cannot be changed. If you want to grant specific permissions to a member, you only need to assign the appropriate role to the member. The following tables list the permissions of each role.

- Supported operations on data objects

Supported operations on data objects

Permission	Developer	Analyst	Viewer
Dataset	Yes	No	No
Supported operations on workbooks	Yes	Yes	Yes
Dashboards	Yes	Yes	Yes
BI Portals	Yes	Yes	Yes

- Supported operations on data

Supported operations on data

Permission	Developer	Analyst	Viewer
Create a data source	Yes	No	No
Modify a data source	Developers can modify only data sources created by themselves.	No	No
Delete a data source	Developers can only delete data sources created by themselves.	No	No
Use a data source	Yes	No	No
Create a dataset	Yes	No	No
Modify a dataset	Developers can modify only datasets created by themselves.	No	No
Delete a dataset	Developers can only delete datasets created by themselves.	No	No
Use a dataset	Yes	Yes	No

- Supported operations on workbooks

Supported operations on workbooks

Permission	Developer	Analyst	Viewer
Create a workbook	Yes	Yes	No
Modify a workbook	Developers can modify only workbooks created by themselves.	Analysts can modify only workbooks created by themselves.	No
Delete a workbook	Developers can only delete workbooks created by themselves.	Analysts can only delete workbooks created by themselves.	No
Preview a workbook	Yes	Yes	Yes
Share a workbook	Developers can only share workbooks created by themselves.	Analysts can only share workbooks created by themselves.	No

Permission	Developer	Analyst	Viewer
Reference a workbook	Yes	Yes	No

- Supported operations on dashboards

Supported operations on dashboards

Permission	Developer	Analyst	Viewer
Create a dashboard	Yes	Yes	No
Modify a dashboard	Developers can modify only dashboards created by themselves.	Analysts can modify only dashboards created by themselves.	No
Delete a dashboard	Developers can only delete dashboards created by themselves.	Analysts can only delete dashboards created by themselves.	No
View a dashboard	Yes	Yes	Yes
Share a dashboard	Developers can only share dashboards created by themselves.	Analysts can only share dashboards created by themselves.	No
Reference a dashboard	Yes	Yes	No
Publish a dashboard	Developers can only publish dashboards created by themselves.	Analysts can only publish dashboards created by themselves.	No

- Supported operations on BI portals

Supported operations on BI portals

Permission	Developer	Analyst	Viewer
Create a BI portal	Yes	Yes	No
Modify a BI portal	Developers can modify only BI portals created by themselves.	Analysts can modify only BI portals created by themselves.	No

Permission	Developer	Analyst	Viewer
Delete a BI portal	Developers can only delete BI portals created by themselves.	Analysts can only delete BI portals created by themselves.	No
View a BI portal	Yes	Yes	Yes
Share a BI portal	Developers can only share BI portals created by themselves.	Analysts can only share BI portals created by themselves.	No

7.7.11.3. Differences between a personal workspace and a group workspace

Workspaces of a user are classified into a personal workspace and group workspaces. The personal workspace is automatically created when the user first logs on to the Quick BI console. A group workspace is created for group members to collaborate. Differences between the two types of workspaces are as follows:

- A personal workspace is automatically created when you log on to the Quick BI console for the first time. A group workspace needs to be manually created by an organization administrator.
- You cannot create or delete a personal workspace.
- You cannot add other users to your personal workspace, nor can you collaboratively edit or transfer it.
- A group workspace can be transferred to any member in the group workspace and shared with all members in the organization. A personal workspace can be shared with Apsara Stack tenant accounts.

7.7.12. Create a workspace

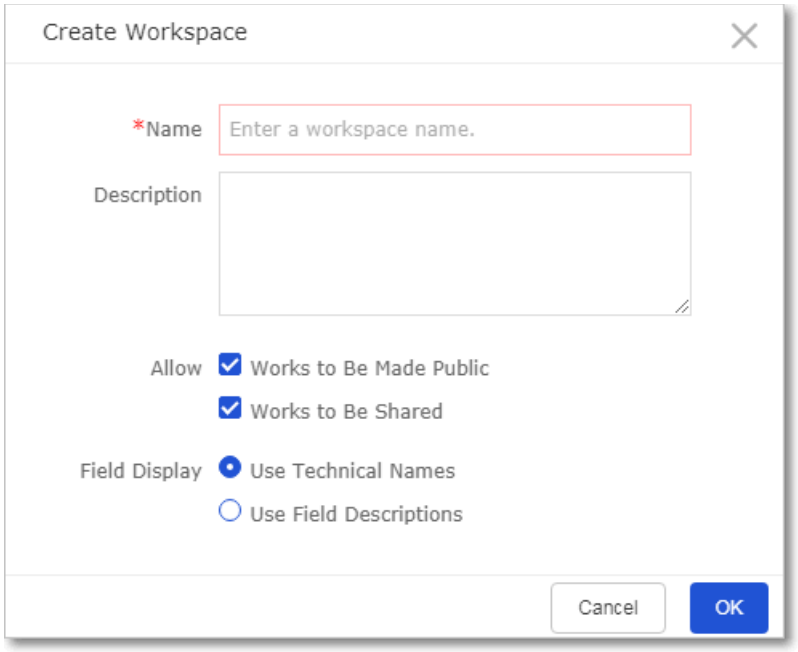
This topic describes how to create a workspace.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. On the Organization page that appears, click **Workspaces** in the left-side navigation pane. Then, click **Create Workspace**.
4. In the Create Workspace dialog box that appears, enter a name for the workspace.



5. Click **OK** to create the workspace.

7.7.13. Modify information about a workspace

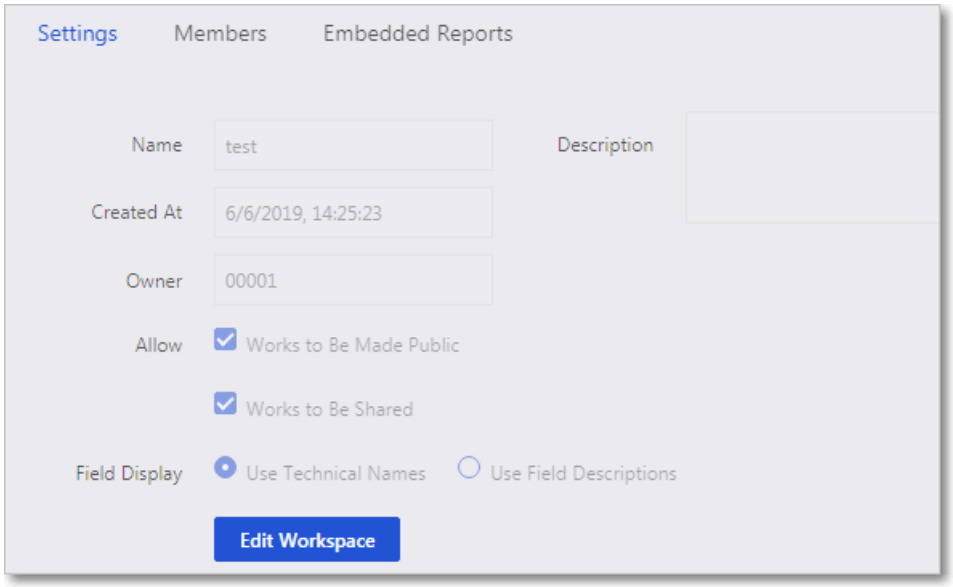
For a personal workspace, only its owner can modify the workspace information. For a group workspace, only its administrators can modify the workspace information.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click **Workspaces**.
4. Click the **Settings** tab.
5. Click **Edit Workspace** and modify the information about the workspace.



6. Click **OK**.

7.7.14. Leave a workspace

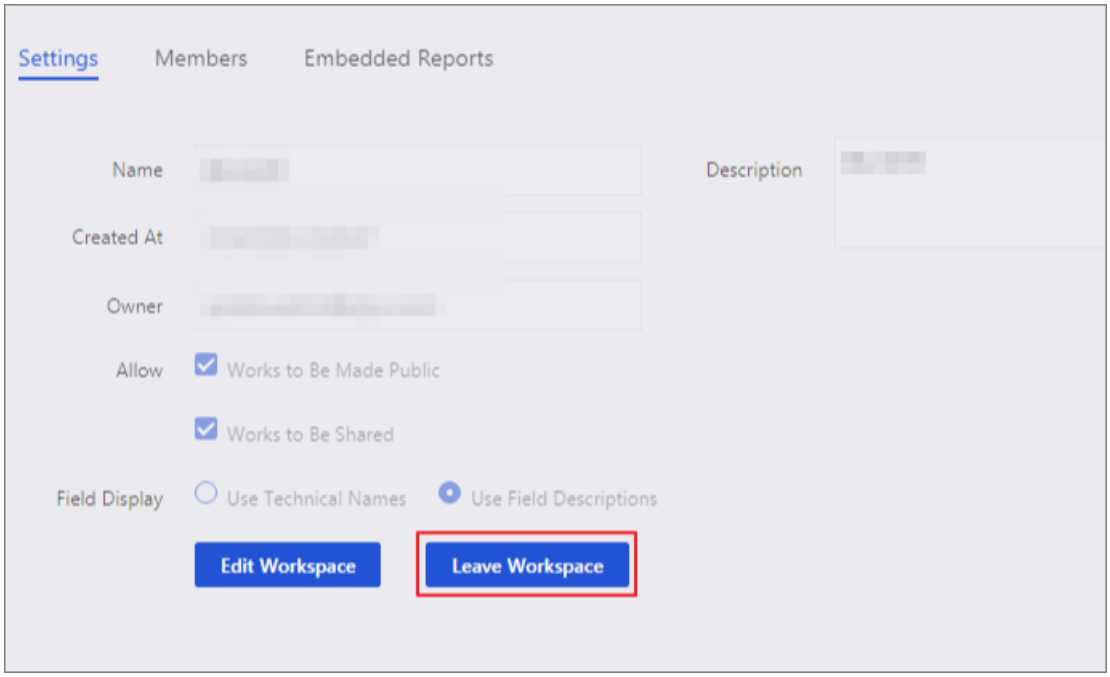
This topic describes how to leave a workspace.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click **Workspaces**.
4. Click the workspace that you want to leave and click the **Settings** tab.
5. Click **Leave Workspace** to leave the current workspace.



7.7.15. Transfer a workspace to another member

This topic describes how to transfer a workspace to another member.

Prerequisites

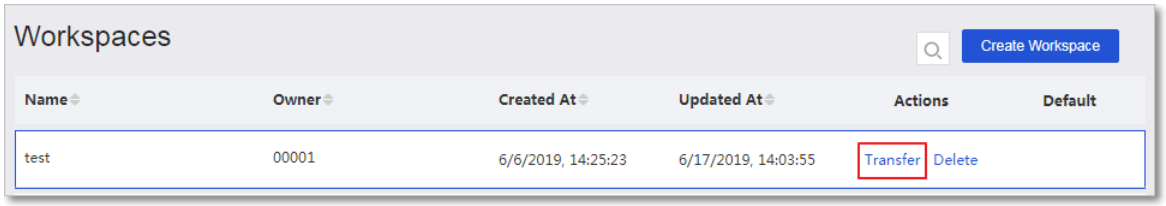
The Quick BI service is purchased.

Context

If you need to remove the owner of a workspace from the organization, you can transfer the workspace to another member. The new owner can be any member in the organization.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click **Workspaces**.
4. Find the workspace that you want to transfer and click **Transfer** in the Actions column.



5. Enter the alias of the new owner and click **OK**.

7.7.16. Delete a workspace

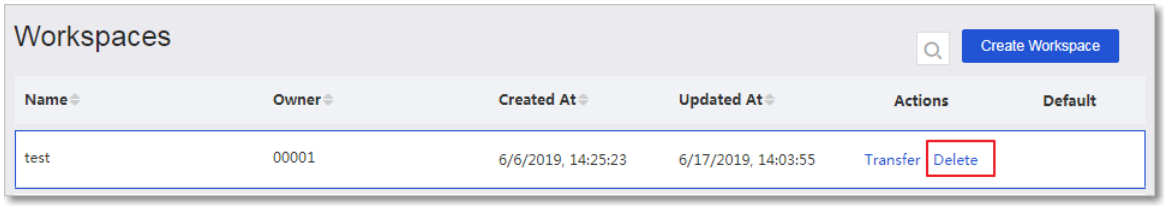
This topic describes how to delete a workspace.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click **Workspaces**.
4. On the Workspaces page, find the workspace that you want to delete and click **Delete** in the Actions column.



7.7.17. Add a member to a workspace

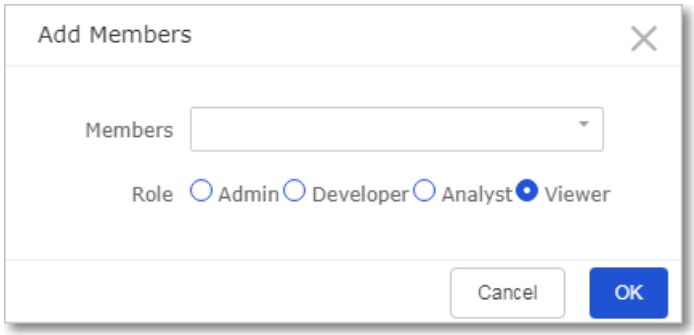
This topic describes how to add a member to a workspace.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console**, or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click Workspaces. On the Workspaces page that appears, click the **Members** tab.
4. Click **Add Members**.
5. Enter the account of the member that you want to add and assign a role to the account.



7.7.18. Edit settings of a workspace member

This topic describes how to edit the settings of a workspace member.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console** or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click Workspaces. On the Workspaces page that appears, click the **Members** tab.
4. Find the member whose settings you want to edit and click **Edit** in the Actions column.
5. Change the settings of the member and click **OK**.

7.7.19. Search for a member in a workspace

You can search for a member on the **Members** tab.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console**, or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click Workspaces.
4. On the Workspaces page that appears, click the **Members** tab. All members in the current workspace are listed on this page.
5. Enter an alias or account in the search box.
6. Click the Search icon to search for the member.

7.7.20. Delete a member from a workspace

This topic describes how to delete a member from a workspace.

Prerequisites

The Quick BI service is purchased.

Procedure

1. Log on to the Quick BI console.
2. On the homepage of the **Quick BI console**, or in a workspace, click the **Settings** icon.
3. In the left-side navigation pane of the Organization page that appears, click Workspaces. On the Workspaces page that appears, click the **Members** tab.
4. Find the member that you want to delete and click **Delete**.
5. From the Specify a new owner drop-down list, select a member as the new owner.

6. Click OK.

7.8. Permissions

7.8.1. Overview of permissions

You can manage Quick BI permissions at the data object and row levels.

Data object-level permission control

The data objects you can manage in Quick BI include data sources, datasets, dashboards, workbooks, and BI portals. You can manage them in personal workspaces or group workspaces. For information about differences between personal workspaces and group workspaces, see Data objects.

Row-level permission control

You do not need to configure row-level permissions for all fields in a dataset. Configure row-level permissions for specific fields based on your business needs.

> **Note** Only datasets in group workspaces support row-level permission control.

7.8.2. Manage data objects

This topic describes how to manage data objects. Data objects include data sources, datasets, dashboards, workbooks, and BI portals.

The following table lists differences between data object management in a personal workspace and that in a workspace, also referred to as a group workspace.

Item	Workspace (Group workspace)	Personal workspace
Permissions	Data objects can be shared or made public. **Note** Data sources and datasets cannot be shared or made public.	Only the owner of a personal workspace can perform operations on data objects.

Item	Workspace (Group workspace)	Personal workspace
Share data objects	Workbooks, dashboards, and BI portals can be shared. Shared data objects are read-only for other Apsara Stack tenant accounts and RAM users. Other Apsara Stack tenant accounts and RAM users cannot modify, delete or save data objects. • Only the data object owner and the administrators of the workspace have the permissions to share the data object. • If the Works to Be Shared check box is cleared in workspace settings, data objects in the workspace cannot be shared. • Data objects can only be shared with Apsara Stack tenant accounts and RAM users of the same organization. Members can view the data objects in the workspace to which the members belong. Data objects can be shared with users in different workspaces within an organization. Authorized users can view the shared data objects in their own personal workspaces.	Workbooks, dashboards, and BI portals can be shared. Shared data objects are read-only for other Apsara Stack tenant accounts and RAM users. Other Apsara Stack tenant accounts and RAM users cannot modify, delete or save data objects. • Only data object owners have the permissions to share the data objects. • Data objects can only be shared with users of Apsara Stack Quick BI. Authorized users can view shared data objects in their own personal workspaces.
Make data objects public	Everyone can access data objects that have been made public by using URLs. We recommend that you do not make the data objects that contain private business data public.	Everyone can access data objects that have been made public by using URLs. We recommend that you do not make the data objects that contain private business data public.

7.8.3. Manage row-level permissions

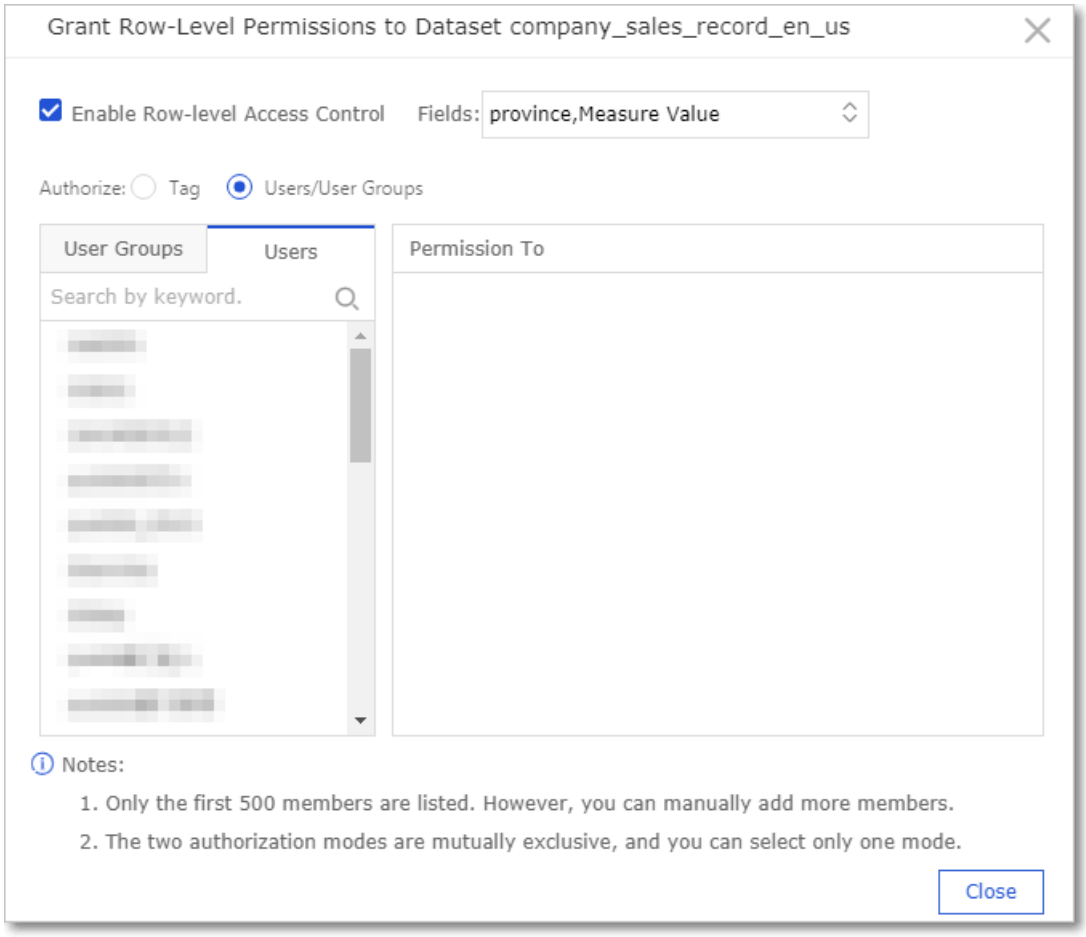
Row-level permissions are managed based on datasets. Quick BI supports the following authorization modes: **user/user group-based authorization** and **tag-based authorization**. **User/user group-based authorization** is ideal for organizations with a small number of members. **Tag-based authorization** is ideal for organizations with a large number of members.

User/user group-based authorization

1. Log on to the Quick BI console
2. Select a workspace. For information about how to create a workspace, see Create a workspace.
3. In the left-side navigation pane of the Workspace page, click **Datasets**.

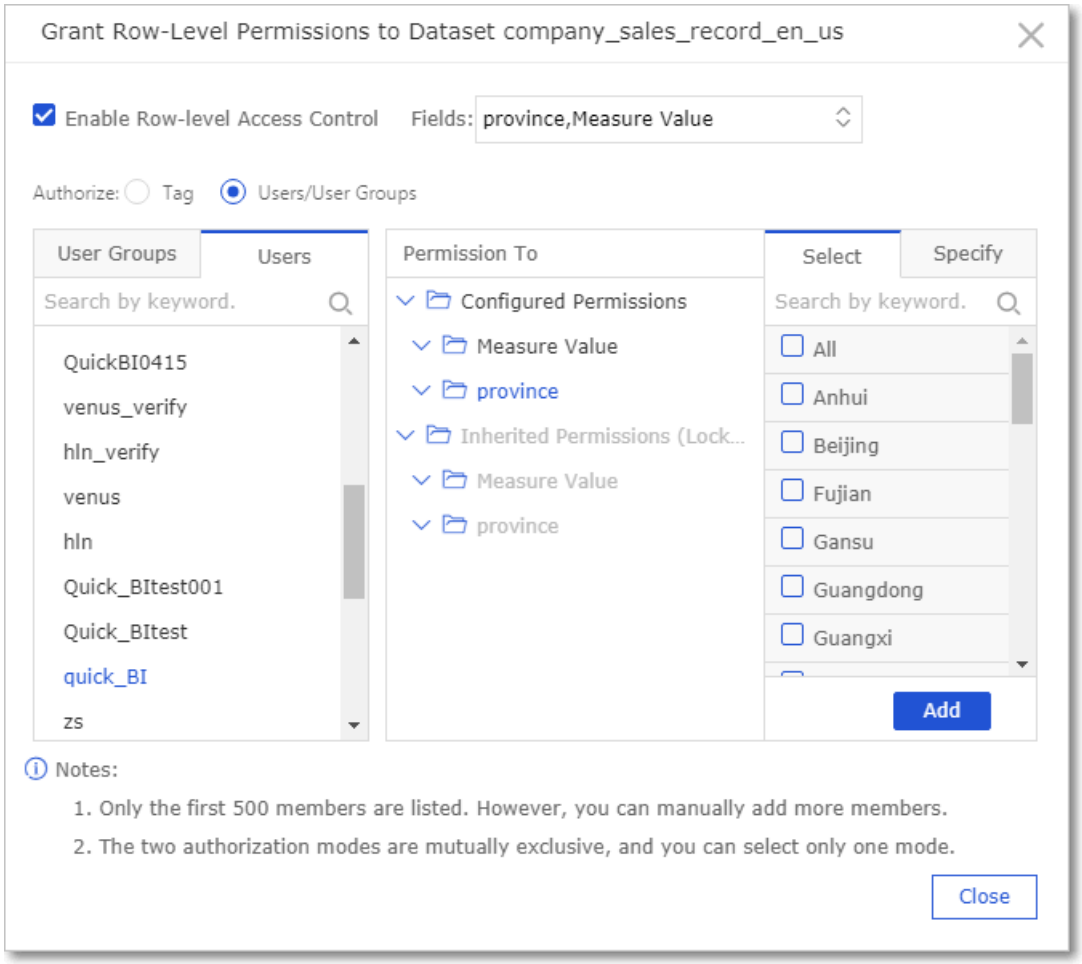
> **Note** Datasets in a personal workspace do not support row-level permission control.

4. Find the target dataset. Click the **More** icon in the Actions column or right-click the dataset.
5. Select **Grant Row-Level Permissions**.
6. On the Grant Row-Level Permissions page, select **Enable Row-level Access Control** and select **Users/User Groups** for Authorize.
7. Click the drop-down arrow of Fields. Select the fields that the authorization is based on, such as province and Measure Value, as shown in the following figure.



The values of the **Measure Value** field are the measures in the dataset. By granting row-level permissions based on the Measure Value field, you can specify the measures available to different users.

8. In the **Permission To** section, click the **province** field. A list of provinces appear.
9. Select a member and choose values of the **province** field to grant permissions to the member, as shown in the following figure.



In this example, the member can view the data of Shanghai and Yunnan.

> **Note** If you grant permissions based on a field of a dataset, you must specify whether all members of the workspace have permissions to access the dataset. Otherwise, when other users attempt to access reports created based on the dataset, the system denies the access requests by default

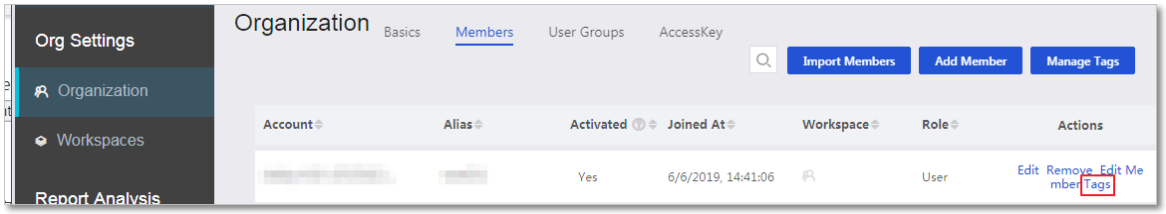
10. Click **Add** to complete the authorization.

Tag-based authentication

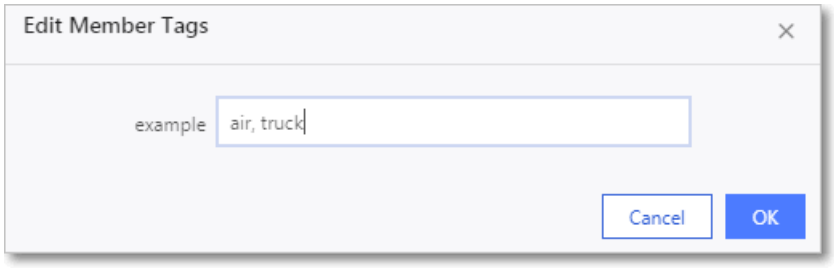
Scenario: Users can only access the data rows in the company_sales_record dataset with shipping_type set to truck and air.

Set member tags

1. In the target workspace, click the **Settings** icon.
2. On the **Settings** page, find the member you want to authorize and click **Edit Member Tags** in the Actions column.



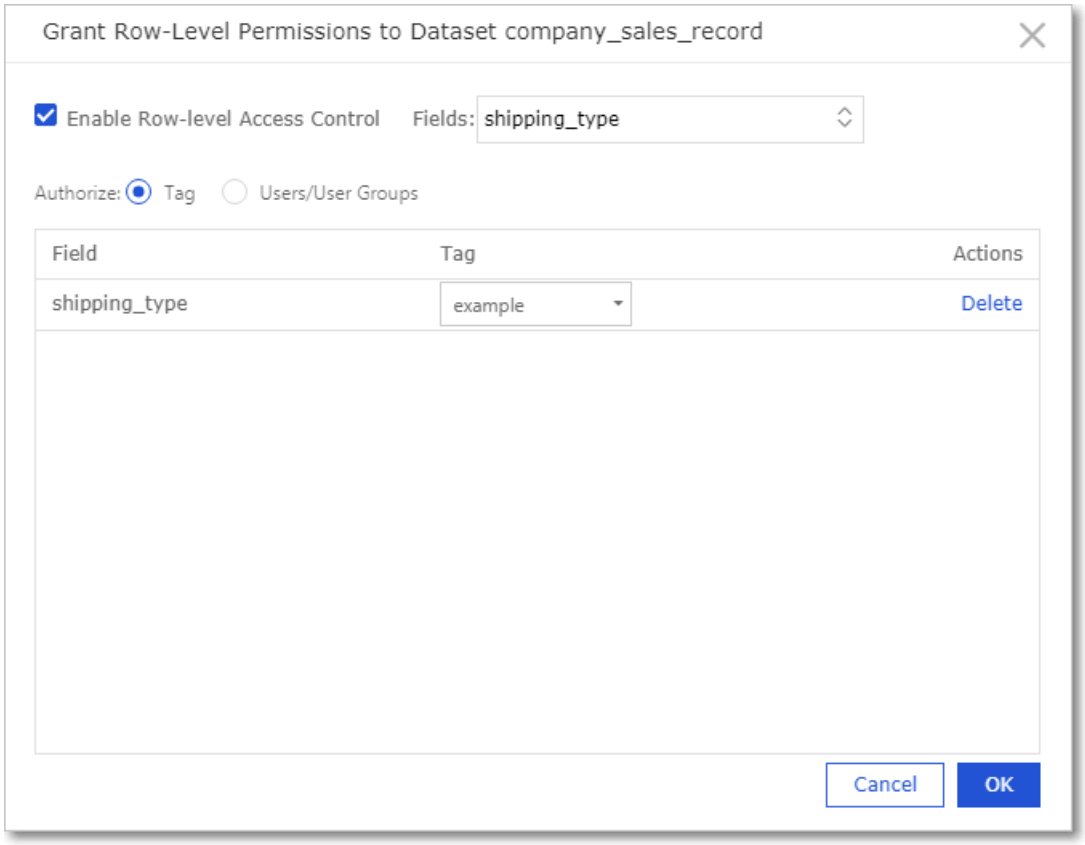
3. In the **Edit Member Tags** dialog box that appears, set the value of the **example** tag to air, truck and click **OK**.



After you set the member tag, you must specify the tag in the Grant Row-Level Permissions dialog box.

Set tag-based authorization

1. Find the company_sales_record dataset. Click the **More** icon in the Actions column or right-click the dataset.
2. Select **Grant Row-level Permissions**.
3. On the Grant Row-Level Permissions page, select **Enable Row-level Access Control** and select **Tag** for Authorize.
4. From the **Fields** drop-down list, select **shipping_type**. Select **example** in the **Tag** column, and click **OK** to complete the settings.



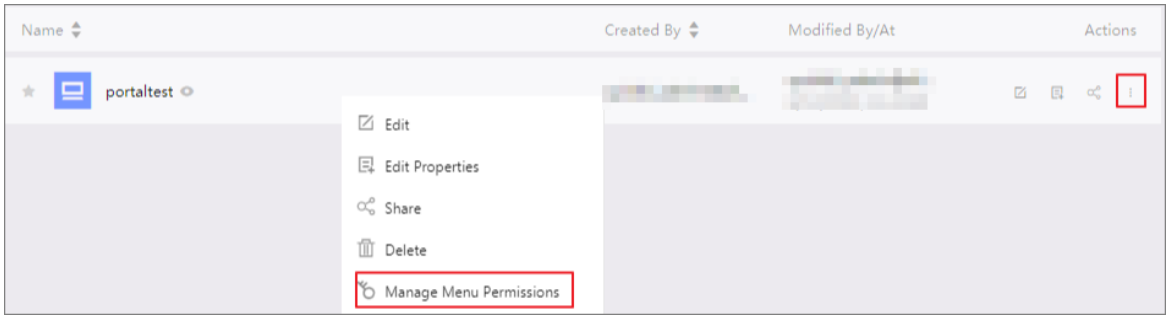
After tag authorization is complete, the user can only access data rows with **shipping_type** set to air or truck.

7.8.4. Configure menu permissions for a BI portal

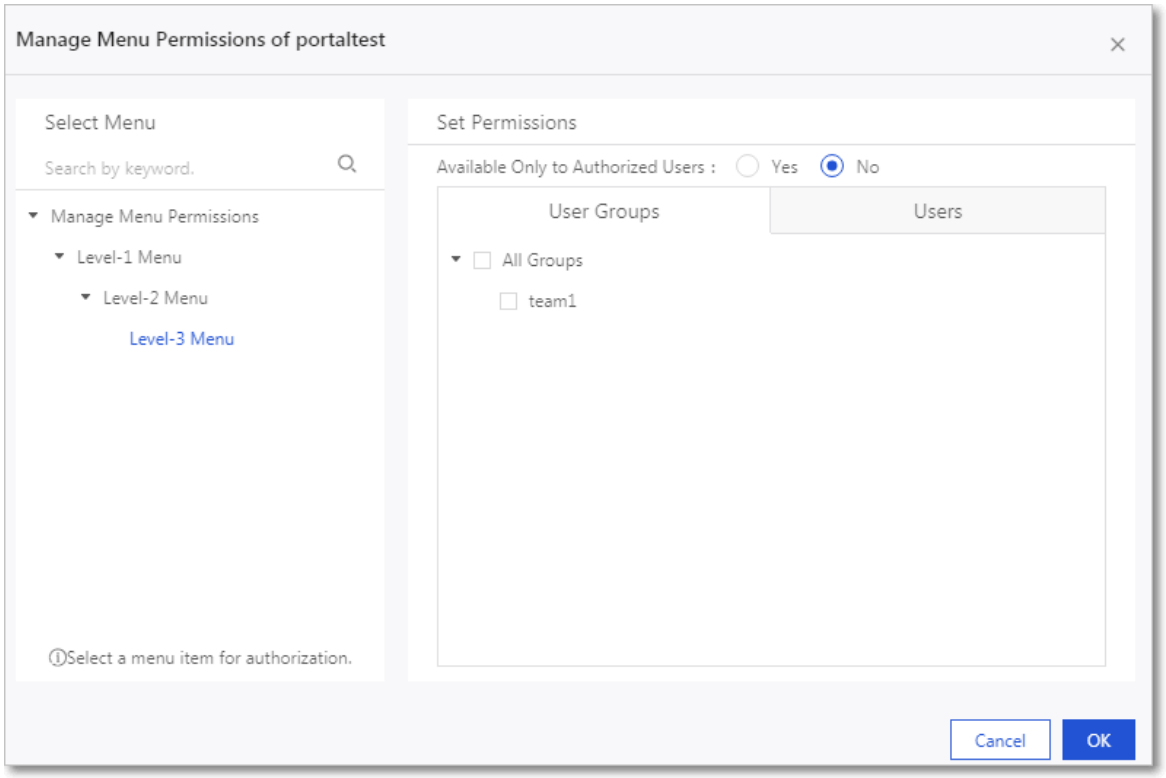
Workspace administrators can configure permissions on viewing BI portal menus in a workspace, also referred to as a group workspace.

The permissions can be granted to user groups or individual users by performing the following steps:

1. Log on to the Quick BI console.
2. Select a workspace. For information about how to create a workspace, see Create a workspace.
3. In the left-side navigation pane of the Workspace page, click **BI Portals**.
4. On the BI Portals page, select the BI portal that you want to configure the menu permissions, click the **More** icon in the Actions column or right-click the BI portal, and select **Manage Menu Permissions**.



5. In the **Manage Menu Permissions** dialog box that appears, select the menu for which you want to manage the permissions, specify whether the menu is available only to authorized users, and select the users or user groups that you want to authorize.



> **Note** The values of **Available Only to Authorized Users** are described as follows:
>
> - Yes: Only authorized users and user groups can access this menu.
> - No: All users and user groups can access this menu.

6. Click **OK**.

7.8.5. Share a data object in a personal workspace

Only the owner of a data object has the permissions to share the data object.

Prerequisites

The Quick BI service is purchased.

Context

In a personal workspace, you can share **workbooks**, **dashboards**, and **BI portals**. Shared data objects are read-only for other Apsara Stack tenant accounts and RAM users. Other Apsara Stack tenant accounts and RAM users do not have the permissions to modify, delete, or save the data objects.

Members with whom a data object is shared can log on to the Quick BI console and view the data object on the **My Items** page.

This topic uses a dashboard as an example to describe how to share data objects in a personal workspace.

Procedure

1. Log on to the Quick BI console.
2. In the left-side navigation pane of the Workspace page, click **Dashboards**.
3. On the Dashboards page, select the dashboard that you want to share with others and click the **Share** icon in the Actions column.
4. Enter the account of the user that you want to share the dashboard with, and specify an expiration date.
5. Click **Save** to share the dashboard.

7.8.6. Share a data object in a workspace

In a workspace, you can share **workbooks**, **dashboards**, and **BI portals**. Shared data objects are read-only for other Apsara Stack tenant accounts and RAM users. Other Apsara Stack tenant accounts and RAM users do not have the permissions to modify, delete, or save the data objects.

Prerequisites

The Quick BI service is purchased.

Context

Only the owner of a data object or administrators of the workspace have the permissions to share the data object. Data objects can only be shared with Apsara Stack tenant accounts and RAM users within the same organization.

If the Works to Be Shared check box is cleared for a workspace, the data objects in the workspace cannot be shared.

This topic takes a dashboard as an example to describe how to share data objects in a workspace.

Procedure

1. Log on to the Quick BI console.
2. Select a workspace.
3. In the left-side navigation pane of the Workspace page, click **Dashboards**.
4. On the Dashboards page, select the dashboard you want to share with others and click the **Share** icon in the Actions column.
5. Enter the alias or account of the user that you want to share the dashboard with, and specify an expiration date.
6. Click **Save** to share the dashboard.

7.8.7. Publish data objects that are stored in a personal workspace

You can publish data objects that are stored in a personal workspace. All Internet users can visit the URLs that point to published data objects. We recommend that you do not publish data objects that include sensitive business data.

Prerequisites

You have purchased Quick BI.

Context

In a personal workspace, you can publish **dashboards** and **workbooks**.

This topic takes a dashboard as an example to describe how to publish data objects in a personal workspace.

Procedure

1. Log on to the Quick BI console.
2. Select a workspace.
3. Click **Dashboards** to go to the Dashboards page.
4. Select the target dashboard and click the **More** icon and select Make Public.
5. Specify an expiration date and click **Make Public**.

   A URL is generated and appears in the Make Public dialog box. You can copy and paste the URL into the address bar of your browser, and then access the dashboard by using the URL.

7.8.8. Make a data object in a workspace public

This topic describes how to make a data object in a workspace, also referred to as a group workspace, public. All Internet users can visit public data objects by using the provided URLs. We recommend that you do not publish data objects that contain sensitive business data.

Prerequisites

The Quick BI service is purchased.

Context

In a workspace, you can make **dashboards** and **workbooks** public.

This topic uses a dashboard as an example to describe how to make data objects in a workspace public.

Procedure

1. Log on to the Quick BI console.
2. Select a workspace.
3. In the left-side navigation pane of the Workspace page, click **Dashboards**.
4. On the Dashboards page, select the dashboard that you want to make public, click the **More** icon in the Actions column, and select Make Public.
5. In the Make Public dialog box, specify an expiration date and click **Make Public**.

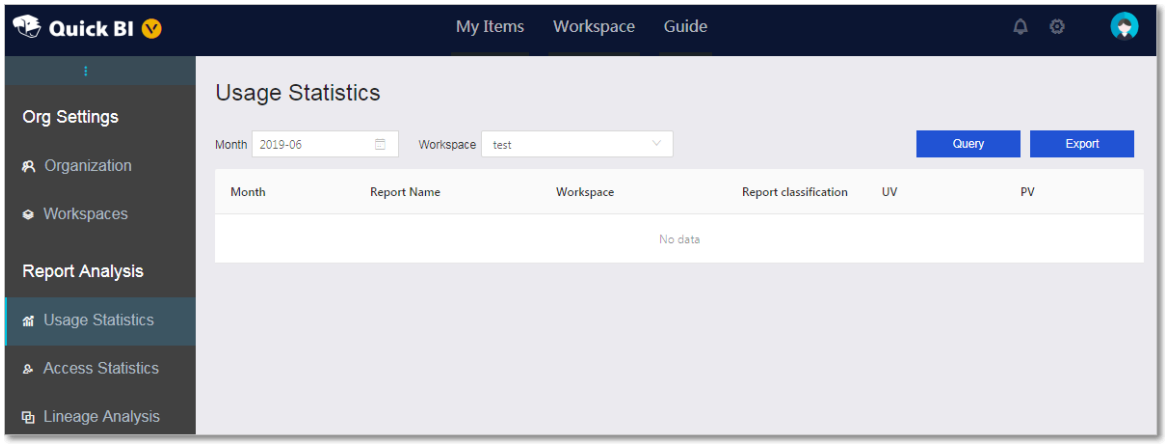
   A URL is generated and appears in the Make Public dialog box. You can copy and paste the URL into the address bar of your browser to access the dashboard.

7.9. Report statistics

7.9.1. Usage statistics

Usage statistics allow you to track the Unique Visitor (UV) and Page View (PV) values of a specific report in a specific month.

1. On the homepage of the Quick BI console, click the **Settings** icon.
2. In the left-side navigation pane, click **Usage Statistics**.
3. On the Usage Statistics page, select a month and workspace and click **Query**.

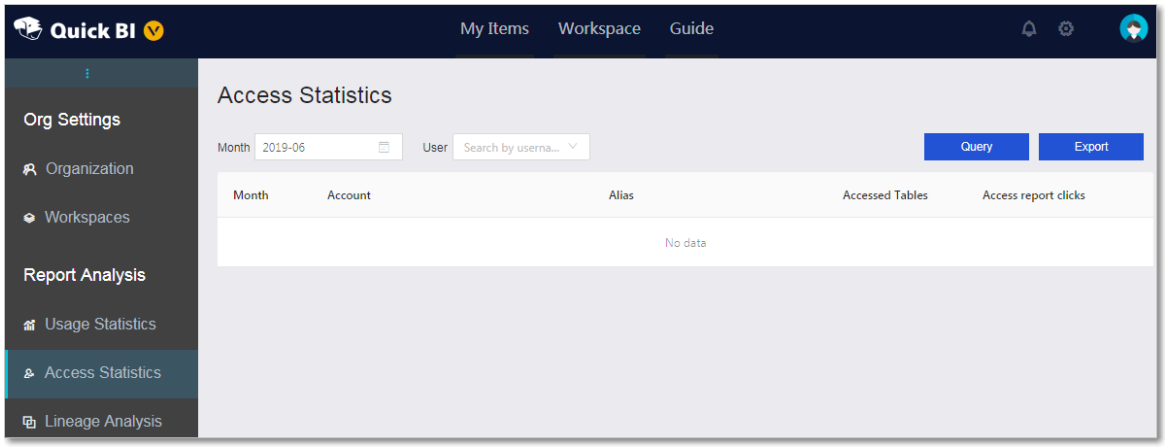


4. Click **Export** to export the statistics data to a local device in an Excel file.

7.9.2. Access statistics

Access statistics allow you to track the number of reports that you have accessed in a specific month and the number of clicks on a specific report.

1. On the homepage of the Quick BI console, click the **Settings** icon.
2. In the left-side navigation pane, click **Access Statistics**.
3. On the Access Statistics page, select a month, enter the member alias, and click **Query**.

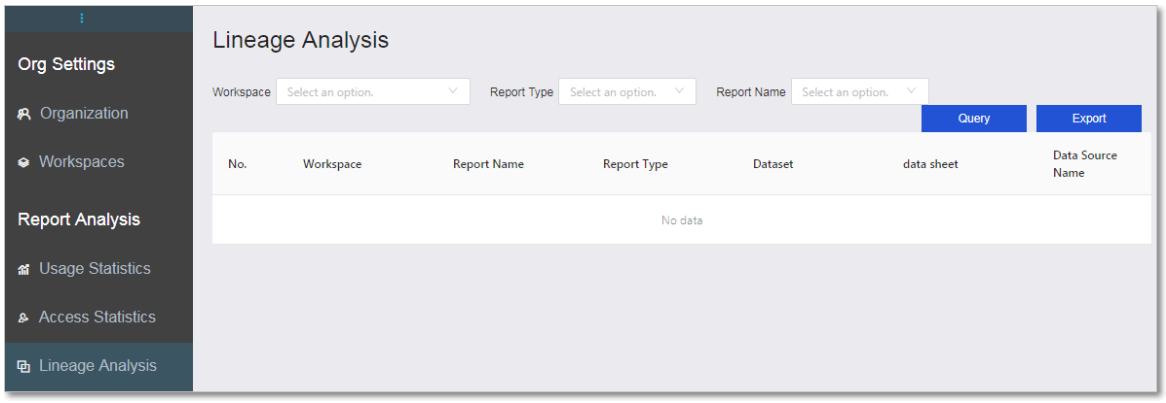


4. Click **Export** to export the statistics data as an Excel file to a local device.

7.9.3. Lineage analysis

Lineage analysis allows you to query information about a specific report, such as the workspace, report type, dataset, chart, and data source name.

1. On the homepage of the Quick BI console, click the **Settings** icon next to the profile picture.
2. In the left-side navigation pane, click **Lineage Analysis**.
3. On the Lineage Analysis page, select the workspace, report type, and report name of the report you want to query, and click **Query**.



4. Click **Export** to export the analysis results as an Excel file to a local device.

8.Graph Analytics

8.1. What is Graph Analytics?

Graph Analytics is a visual analysis platform for relationship networks. Graph Analytics is widely used in Alibaba Group and Ant Financial for risk control including anti-fraud, anti-theft, and anti-money laundering solutions. Graph Analytics provides solutions for multiple industries, including public security protection, taxation, customs, banking, insurance, and the Internet.

Graph Analytics is designed to facilitate multi-source data integration, computing applications, visual analytics, and intelligent businesses. Based on relationship networks, Graph Analytics can visualize the properties of objects and reveal the relationship among objects.

Graph Analytics provides features including relationship networks, search networks, intelligent networks, information cubes, intelligent judgement, collaboration and sharing, and dynamic modeling. It visualizes data and integrates machine computing capabilities with human cognition. This allows you to gain insight into massive data and obtain information and knowledge directly and efficiently.

8.2. Quick Start

8.2.1. Log on to Administration Console of Graph Analytics

Administration Console is the data configuration platform for Graph Analytics. In Administration Console, you can configure data sources, objects, links, events, and other advanced configuration items. Before you use Graph Analytics, you must log on to Administration Console to perform the related configurations.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

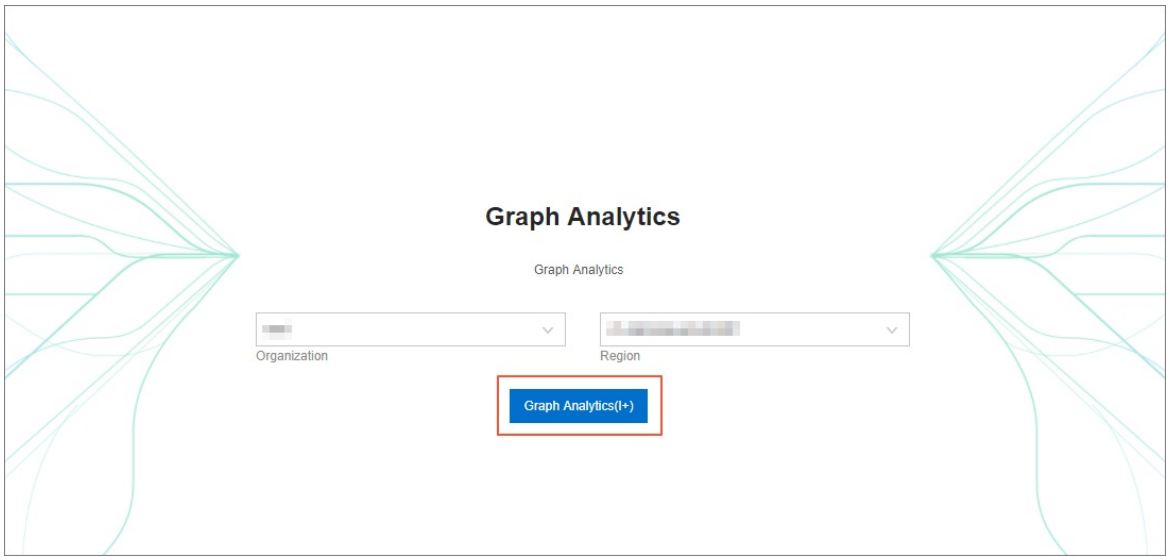
Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

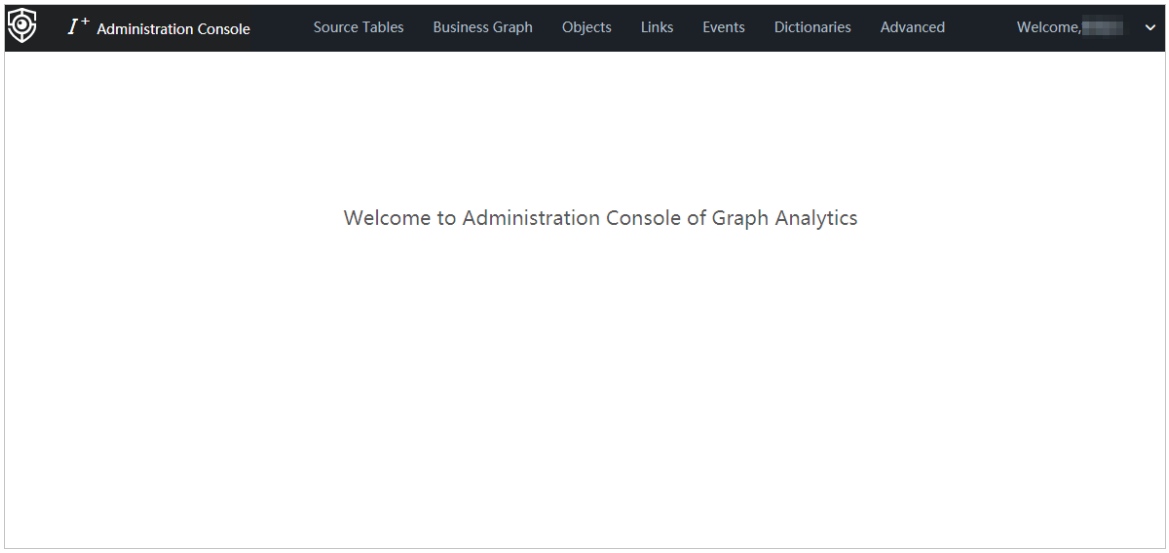
> **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters.
> - Digits.
> - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Product** > **Big Data** > **Graph Analytics**.
5. On the Graph Analytics page that appears, specify **Organization** and **Region**, and click **Graph Analytics(I+)** to redirect to the homepage of Analytics Workbench.



> **Note** You cannot use the root organization for the redirection.

6. Move your pointer over the username in the upper-right corner and select **Administration Console** to redirect to Administration Console of Graph Analytics.



8.2.2. Create data sources

Before you perform a relationship analysis, you must integrate data that you want to analyze, typically databases, into Graph Analytics. These databases will be used as data sources. In Graph Analytics, every data source is unique and can be added only once.

Prerequisites

- You have an account and password for Graph Analytics and are authorized to perform operations on Administration Console of Analytics Workbench.
- You have obtained the IP address, user name, password, port number, and other information of an accessible data source.

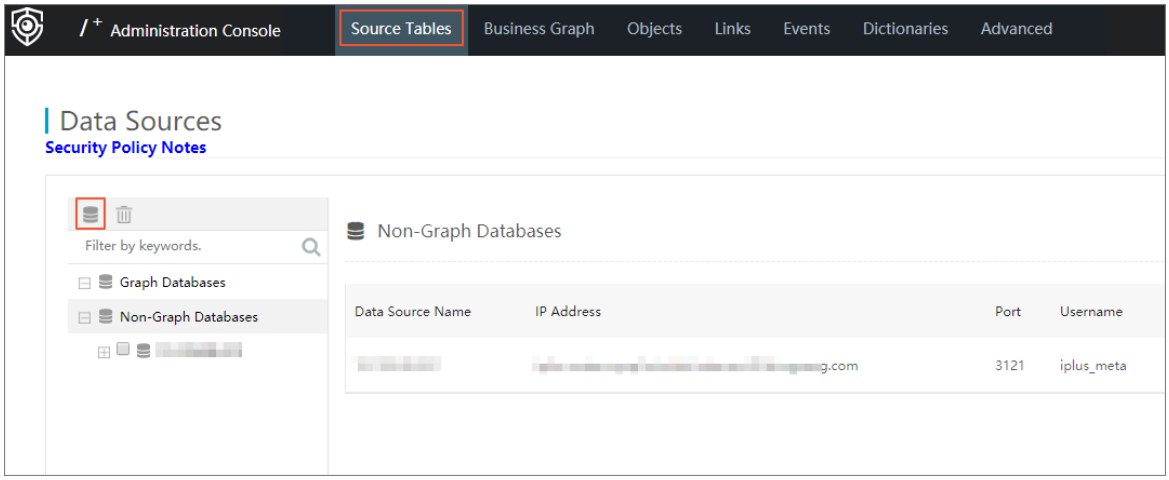
Context

A data source is one of the entries for new objects, links, and events. It is also the only entry for objects, links, and events to map to a data table. Objects, links, and events that are created on the **Object Information**, **Link Information** and **Event Information** pages are logical business objects, links, and events without mappings.

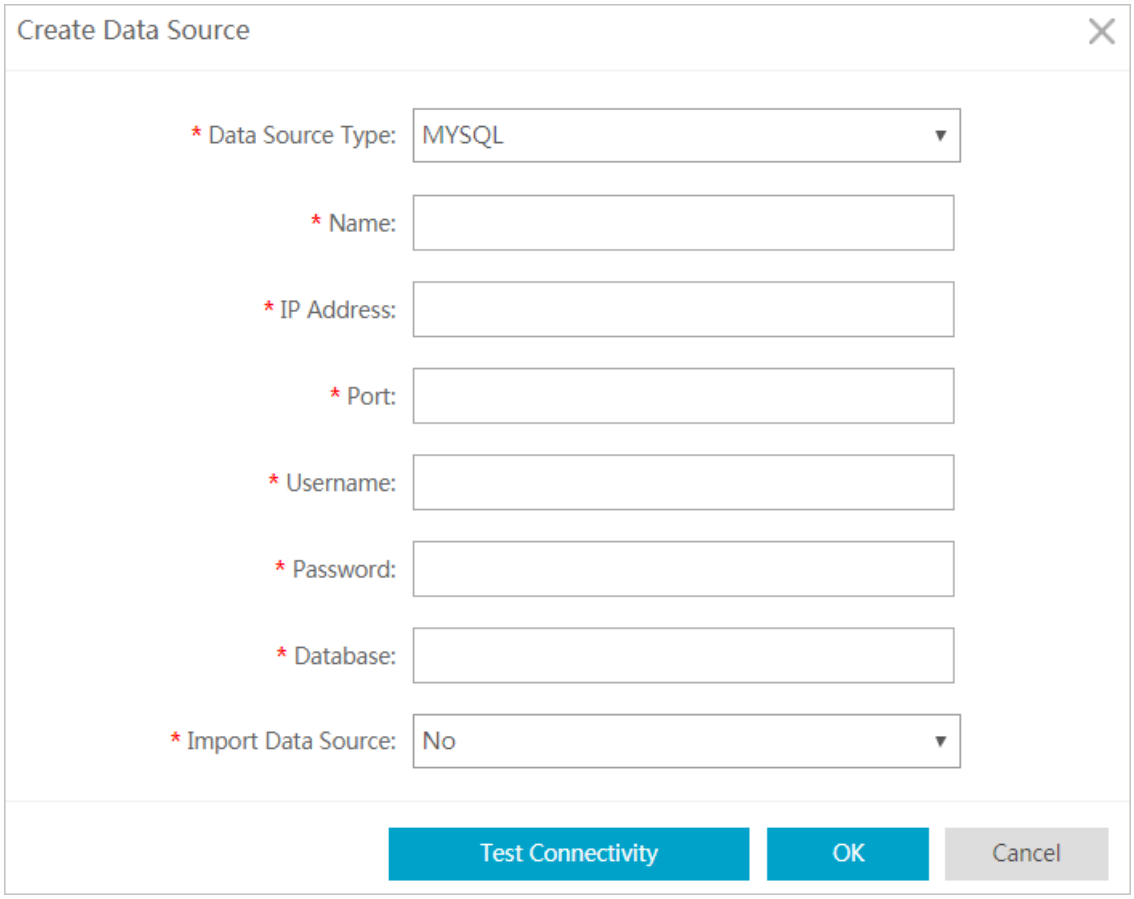
> **Note** Objects, links, and events created for the data source take effect only after you log on again.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Source Tables** to redirect to the **Data Sources** page.



3. On the **Data Sources** page, click the **Create Data Source** icon in the left-side navigation pane. The **Create Data Source** dialog box appears.



4. In the **Create Data Source** dialog box that appears, specify the data source parameters as needed.

Data source parameters

Parameter	Description
Data Source Type	The data source type you want to select. Supported data source types include: **MYSQL**, **ORACLE** , **RDS** , and **GREENPLUM**.

Parameter	Description
Name	The name of the data source. It can be user-defined.
IP Address	The IP address or domain name of the data source.
Port	The port number of the data source.
Group	The department or group to which the data source belongs.
Username and Password	The username and password that are used to connect to the data source.
Database	The name of the database that functions as the data source.
Import Data Source	If Data Source Type is not set to **ORACLE**, you must specify this parameter. You can import data into Analytics Workbench only after you have specified this parameter.

5. After you have configured the preceding parameters, click **Test Connectivity** to check whether the data source can be connected.

   If the data source is connected properly, a message appears, indicating that the test succeeded. If the data source cannot be connected, check whether the information is correct and the data source itself is functioning properly.

6. After the connectivity test is confirmed as successful, click **OK**.

8.2.3. Create OLEP models for tables

After you add a data source, you must create object, link, event, and property (OLEP) models for the tables in the data source as needed. Before you configure OLEP tables, prepare the tables for which you will create OLEP models, the columns of each table, and the business models you want to configure. Referenced tables cannot be deleted.

Prerequisites

You have created an accessible data source.

Context

OLEP models include the following types of mappings: table-to-object mappings, table-to-link mappings, and table-to-event mappings. You can add objects, links, and events when you create OLEP models. After that, you can view and configure these objects, links, and events on the **Object Information**, **Link Information**, and **Event Information** pages, respectively. You can configure these items to reflect your business semantics. An OLEP table serves as a source for configuring objects, first-degree links, and events. A table can be mapped to multiple objects, links, and events.

Procedure

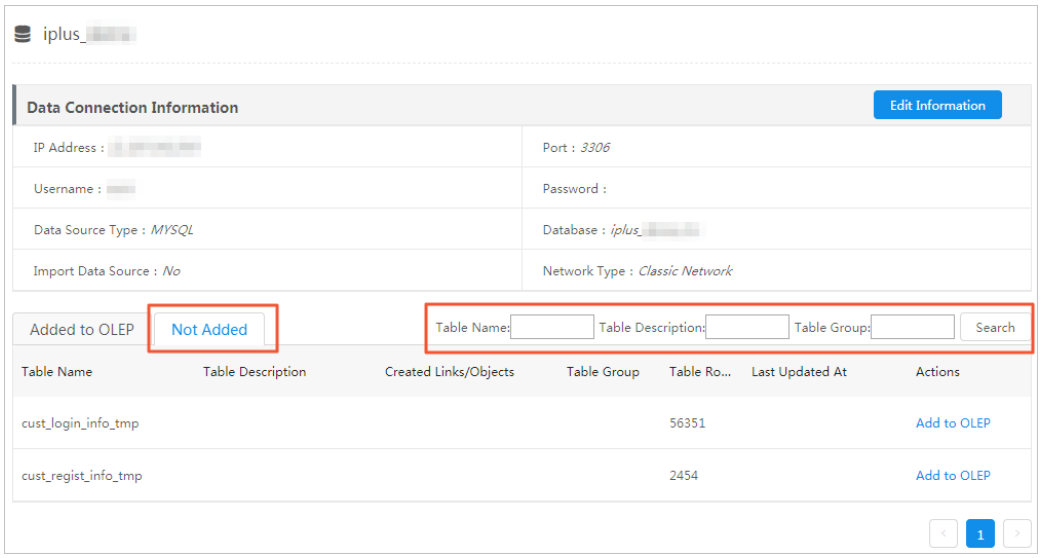
1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar. The **Data Sources** page appears.
3. Click a data source in the left-side navigation pane. The data source details are displayed

on the right side of the page.

4. Click the **Not Added** tab. All tables that have no OLEP models are displayed.

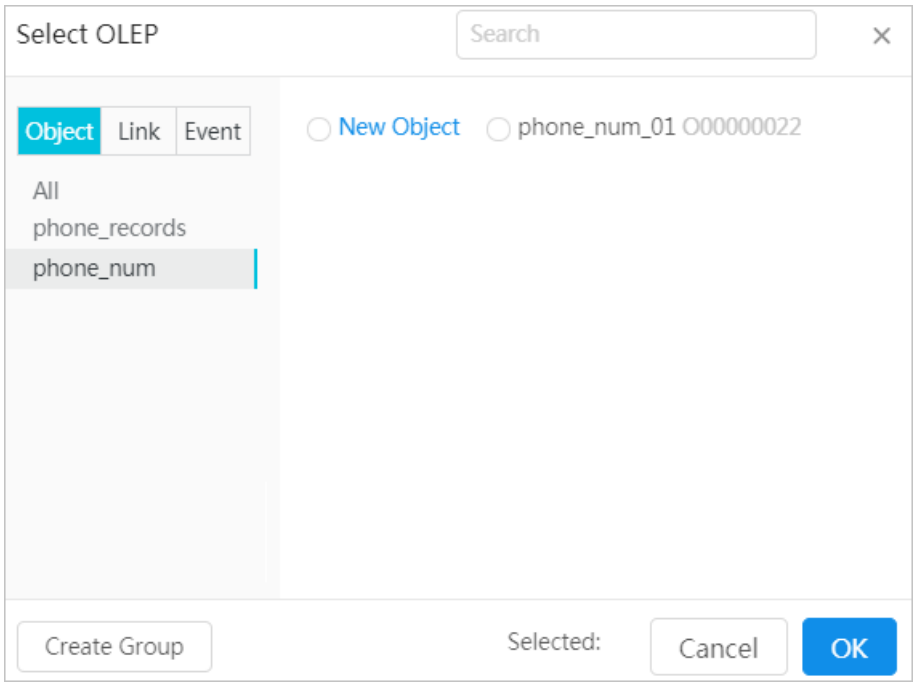
   You can search for a table quickly and accurately by specifying the Table Name, Table Description, or Table Group parameter.

   Tables that have no OLEP models



5. Select a table, and then click **Add to OLEP** in the Actions column. The **Select OLEP** dialog box appears.

   Select OLEP dialog box

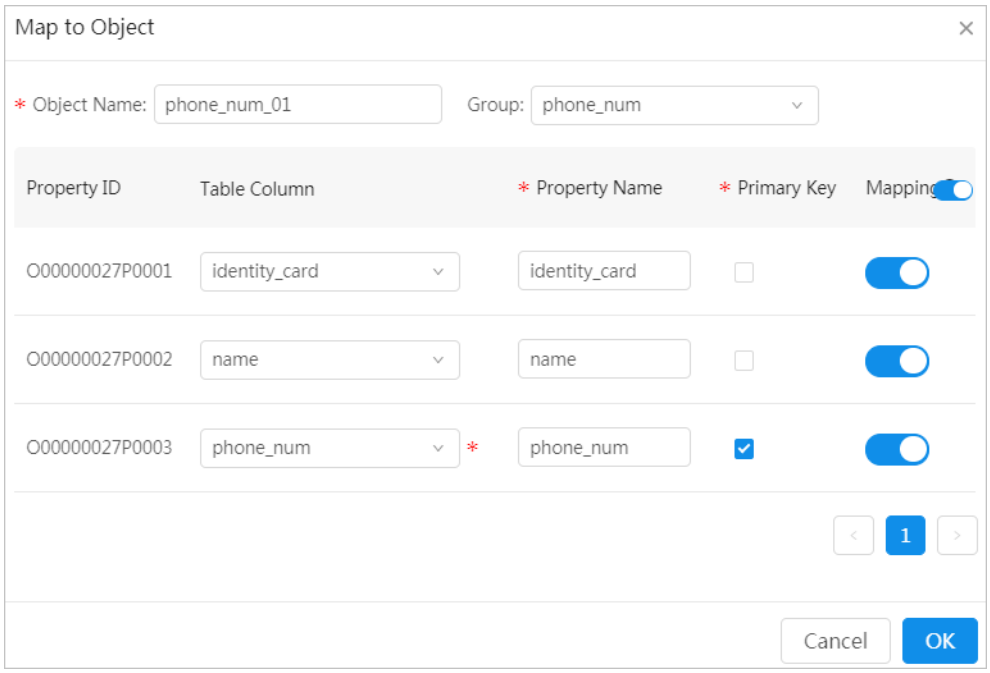


   The **Select OLEP** dialog box contains the **Object**, **Link**, and **Event** tabs which are used to create mappings to objects, links, and events, respectively. For more information about how to create a mapping to an object, link, or event, see Step 6, Step 7, and Step 8.

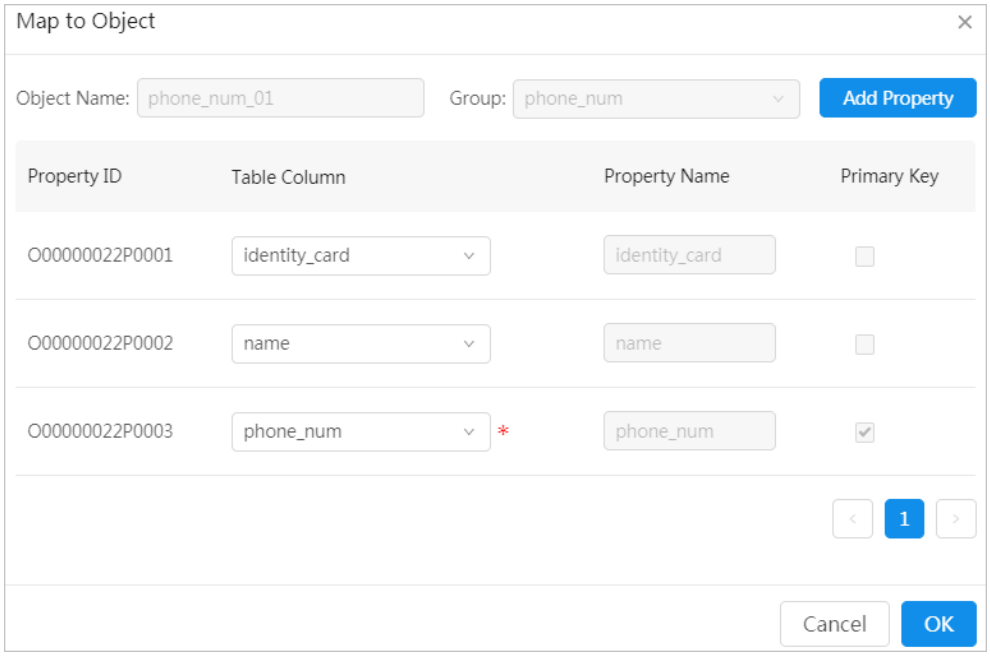
If no existing object groups, link groups, or event groups meet your requirements, click **Create Group** to add an object group, link group, or event group.

6. Map the table to an object.
   i. Select **New Object** or an existing object and then click **OK**. The **Map to Object** dialog box appears.

Map to a newly created object



Map to an existing object



ii. Configure the parameters as needed.

- New object: Configure the parameters based on Parameters used to map the table to a new object.
- Existing object: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns is greater than the number of existing properties of the object, you can click **Add Property** to add new properties.

Parameters used to map the table to a new object

Field	Description
Object Name	The user-defined object name. It must be unique.
Group	The object group to which the object belongs. All available object groups are displayed in the drop-down list.
Property Name	The name of an object property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, property names are displayed instead of the actual table columns that are mapped to the properties.
Mapping	Indicates whether to enable the property mapping.
Primary Key	The property that is used as a primary key. Each primary key uniquely identifies an object. You must configure one or more properties as primary keys for each object. You must enable **Mapping** for primary keys.

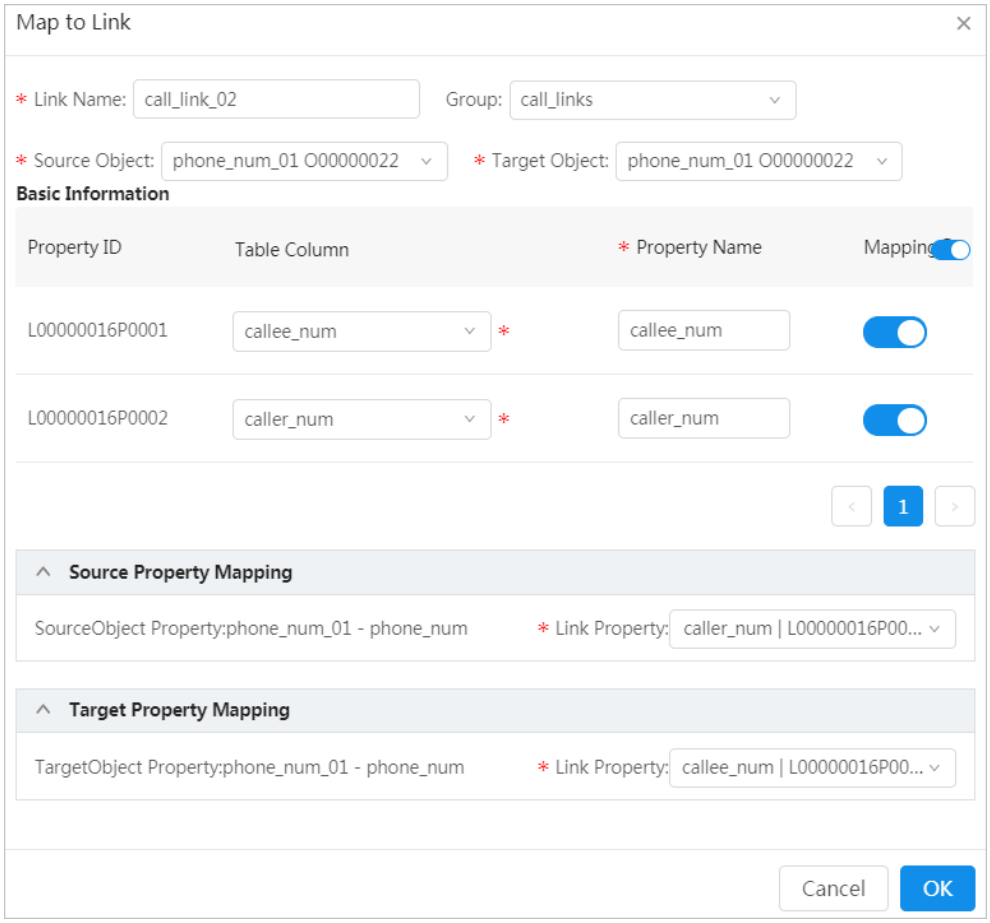
iii. Click **OK**.

7. Map the table to a link.

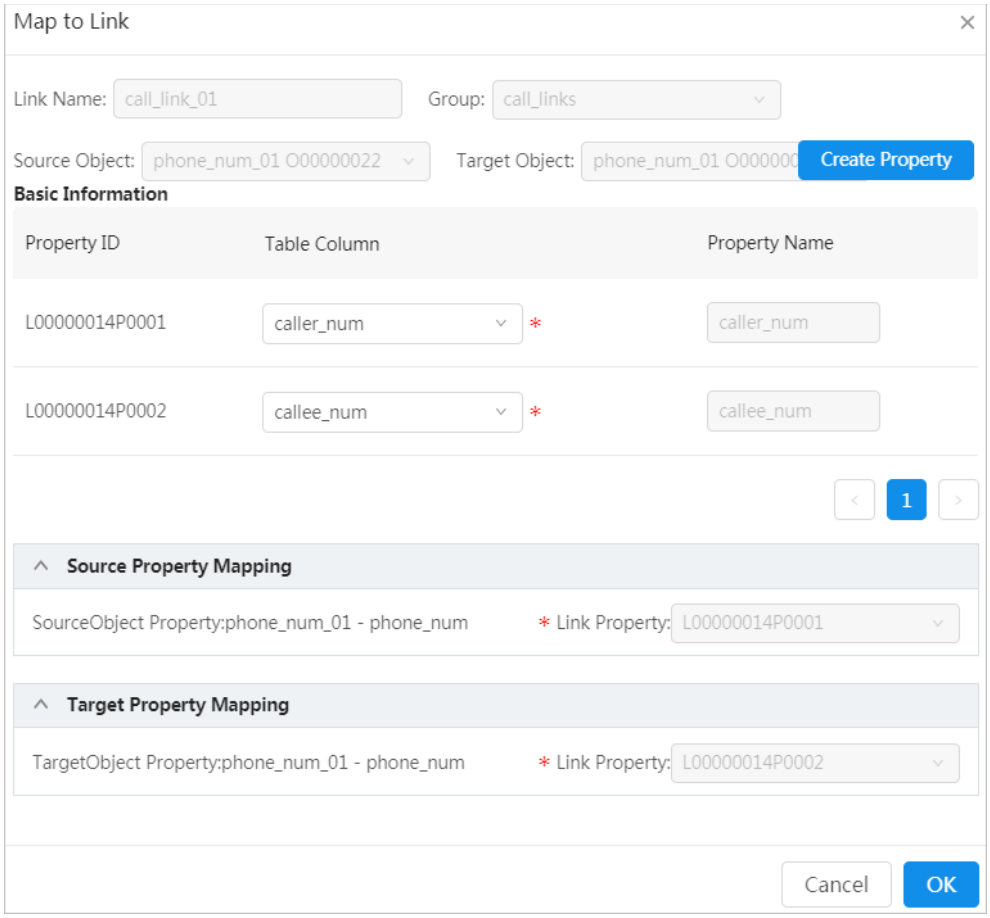
i. Click the **Link** tab. All first-degree links to which the current table has been mapped are displayed.

ii. Select **New Link** or an existing link and then click **OK**. The **Map to Link** dialog box appears.

Map to a newly created link



Map to an existing link


Map to Link
Link Name: call_link_01
Group: call_links
Source Object: phone_num_01 O00000022
Target Object: phone_num_01 O000000
Create Property
Basic Information
Property ID
Table Column
Property Name
L00000014P0001
caller_num
caller_num
L00000014P0002
callee_num
callee_num
1
Source Property Mapping
SourceObject Property:phone_num_01 - phone_num
Link Property: L00000014P0001
Target Property Mapping
TargetObject Property:phone_num_01 - phone_num
Link Property: L00000014P0002
Cancel
OK

iii. Configure the parameters as needed.

- New link: Configure the parameters based on Parameters used to map the table to a new link.
- Existing link: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns is more than the number of existing link properties, you can click **Add Property** to add new link properties.

Parameters used to map the table to a new link

Field	Description
Link Name	The user-defined link name. It must be unique.
Group	The link group to which the link belongs. All available link groups are displayed in the drop-down list.
Source	The source object of the link. You can select an object from the drop-down list. The **Source Property Mapping** parameter is displayed only after you configure the Source Object parameter.
Target	The target object of the link. You can select an object from the drop-down list. The **Target Property Mapping** parameter is displayed only after you configured the Target Object parameter.
Property Name	The name of a link property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, the Property Name values are displayed as the table header in **Details** on the **Graph** page.
Switch	Indicates whether to enable the property mapping.
Link Property in Source Property Mapping	The link property to which a primary key property of the source object is mapped.
Link Property in Target Property Mapping	The link property to which a primary key property of the target object is mapped.

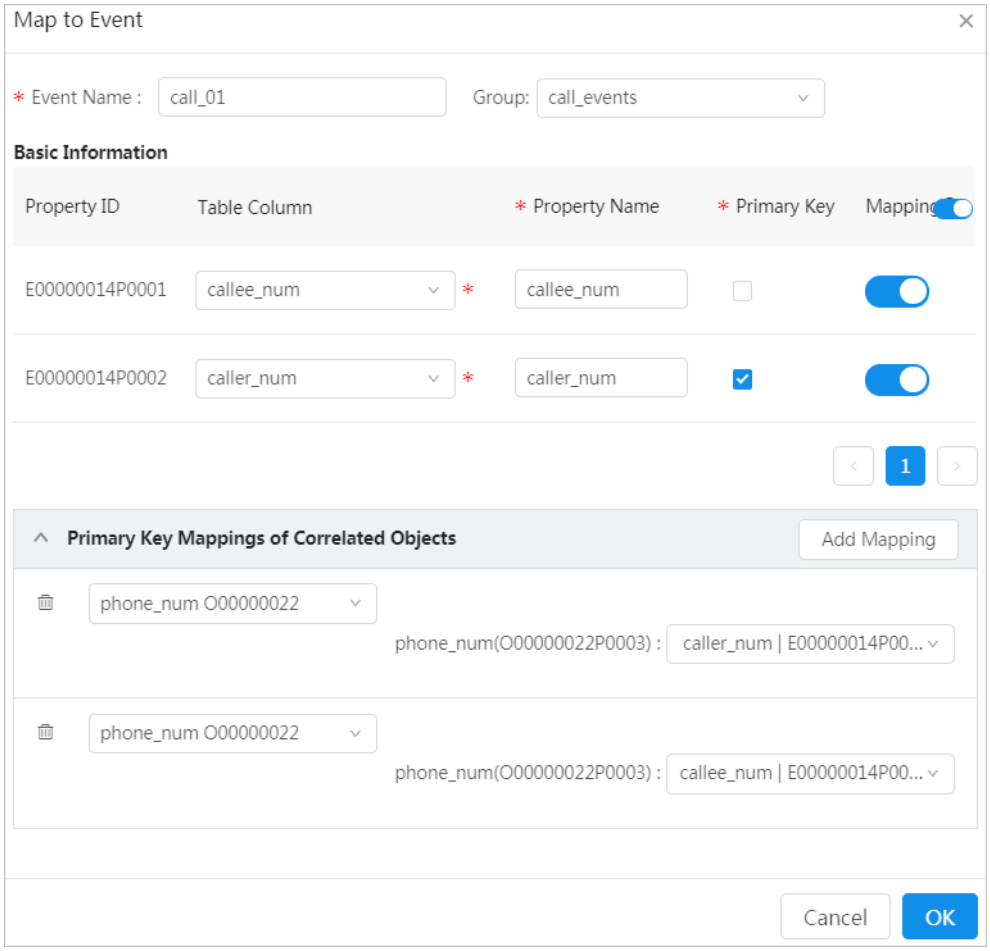
iv. Click **OK**.

8. Map the table to an event.

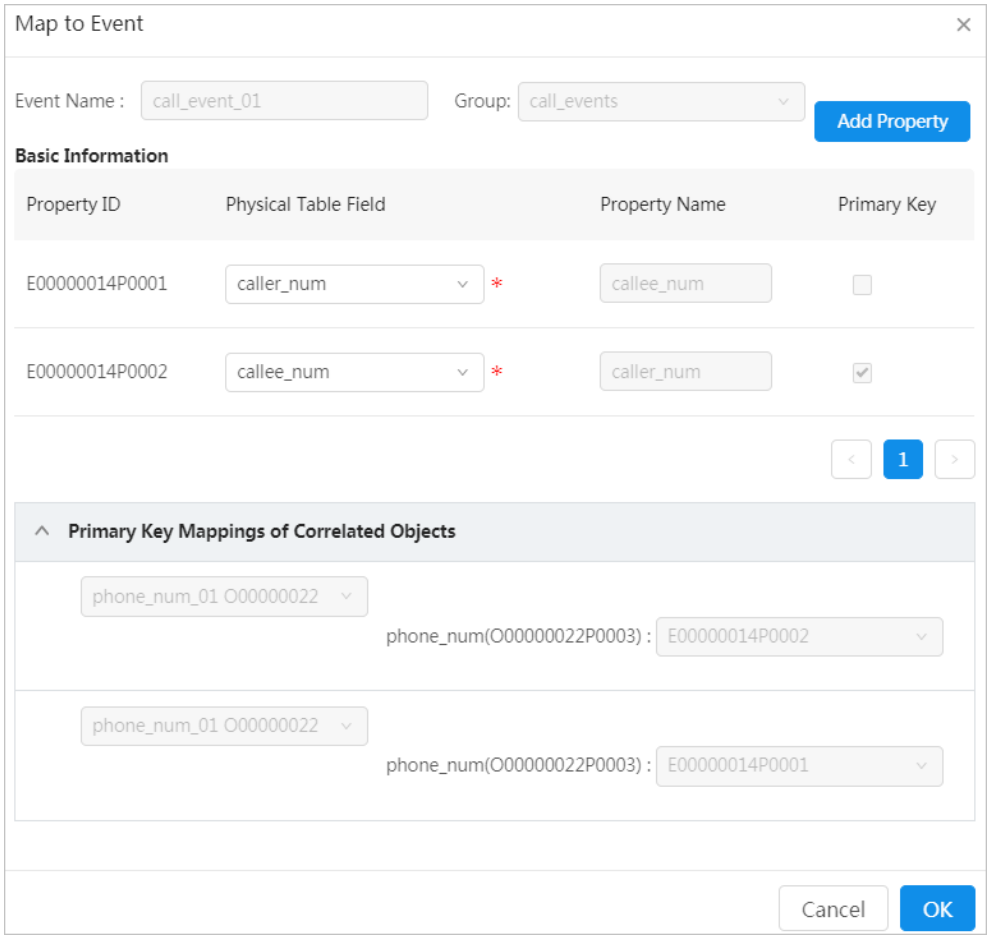
i. Click the **Event** tab. All events to which the current table has been mapped are displayed.

ii. Select **New Event** or an existing event and then click **OK**. The **Map to Event** dialog box appears.

Map to a newly created event



Map to an existing event


Map to Event
Event Name :
call_event_01
Group:
call_events
Add Property
Basic Information
Property ID
Physical Table Field
Property Name
Primary Key
E00000014P0001
caller_num
callee_num
E00000014P0002
callee_num
caller_num
1
Primary Key Mappings of Correlated Objects
phone_num_01 O00000022
phone_num(O00000022P0003) :
E00000014P0002
phone_num_01 O00000022
phone_num(O00000022P0003) :
E00000014P0001
Cancel
OK

iii. Configure parameters as needed.

- New event: Configure the parameters based on Parameters used to map the table to a new event.
- Existing event: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns is greater than the number of existing event properties, you can click **Add Property** to add new event properties.

Parameters used to map the table to a new event

Field	Description
Event Definition Name	The user-defined event name. It must be unique.
Group	The event group to which the event belongs. All available event groups are displayed in the drop-down list.
Property Name	The name of an event property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, the Property Name values are displayed as the table header in **Details** on the **Graph** page.
Switch	Indicates whether to enable the property mapping.
Primary Key	The property that is used as a primary key. Each primary key uniquely identifies an event. You must configure one or more properties as primary keys for each event. **Switch** must be turned on for the properties that are configured as primary keys.
Map Primary Keys to Correlated Objects	The mappings between the primary keys of correlated objects and the event properties. Two or more correlated objects are required. You can click **Add Mapping** to add more necessary mappings between the primary keys of correlated objects and the event properties. You must enable **Mapping** for the event properties to which the primary keys of the correlated objects are mapped.

iv. Click **OK**.

9. After you create OLEP models for the data tables, click the **Added to OLEP** tab to check the results.

8.2.4. Add OLEP table columns

If a data table has been mapped to an object, link, or event, and the table still has unoccupied columns (columns that are not correlated with any object, link, or event), you can add these columns to the existing mappings as needed.

Prerequisites

A data table has been mapped to an object, link, or event, but the table still has unoccupied columns.

Context

Before you configure the OLEP table columns, sort out the columns for which you will create OLEP models and data types of the columns, especially the time columns.

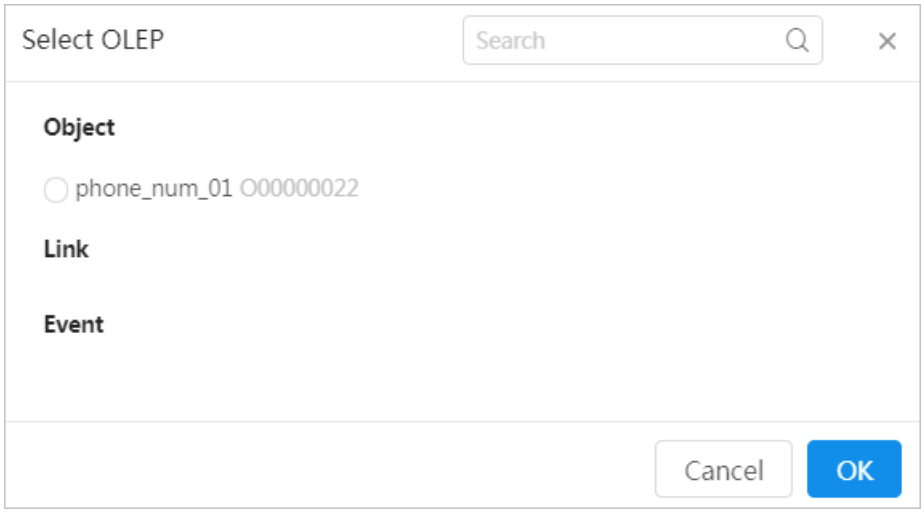
Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar.
3. On the **Data Sources** page, click a data source in the left-side navigation pane, and then click the table to which you want to add the columns.
4. Click the **Columns Not Added** tab in the right-side area. In the **Columns Not Added** tab that appears, click **Add** in the Actions column.

   Columns displayed in the **Columns Not Added** tab are not mapped to any object, link, or event.
5. In the **Select OLEP** dialog box that appears, select the object, link, or event to which the columns map, and then click **OK**.

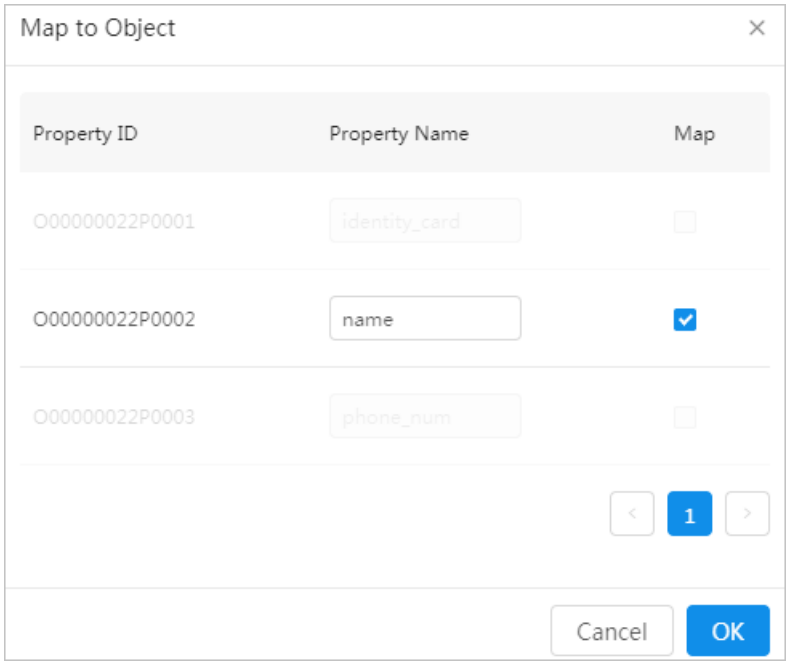
   The **Select OLEP** tab only displays the objects, links, and events that have mapped to the current data table.

   The following example demonstrates how to add columns to an object.

6. In the **Map to Object** dialog box that appears, select the object properties that map to the columns you want to add.

   Object properties that have been mapped to the current data table are dimmed and cannot be operated.



7. After you configure the preceding parameters, click **OK**.

8.2.5. Configure object properties and business parameters

After you add an object, you need to configure the business parameters of the object based on your requirements so that you can view and analyze the object in Analytics Workbench.

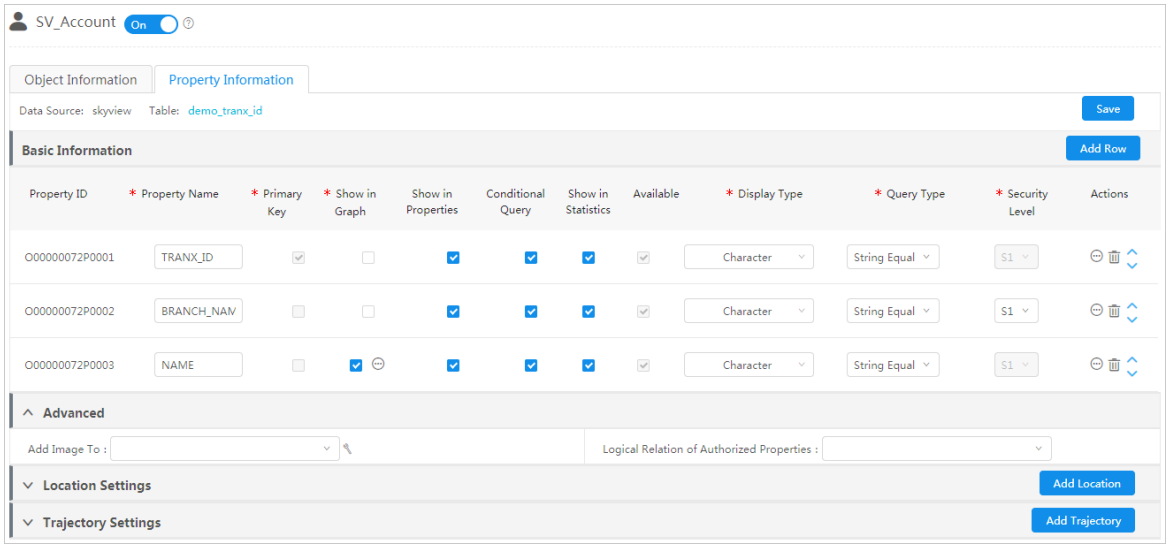
Prerequisites

- You have created a data source. For more information, see Create data sources.
- You have configured mappings between tables and objects, links, or events. For more information, see Create OLEP models for tables.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Objects** on the top of the page.
3. In the left-side navigation pane of the **Object Information** page, click the name of the object you want to configure and then click the **Property Information** tab on the right side.

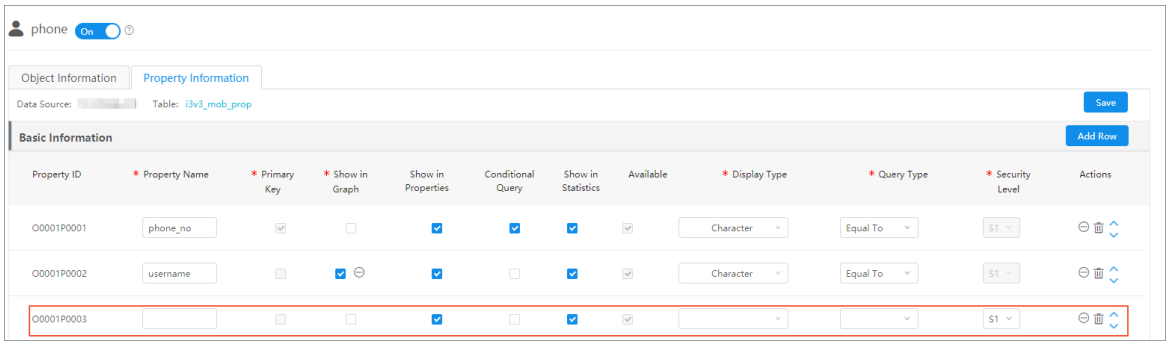
   If this object has been mapped to a physical table in the data source, the **Property Information** tab displays the **Data Source** and **Table** information.



4. Specify parameters in the **Basic Information** section.

   Click **Add Row** to add a property in **Basic information.**

   Parameters in Basic Information describes the parameters in **Basic Information.**



Parameters in Basic Information

Parameter	Description
Property ID	The ID of a property. It is automatically generated.
Property Name	The property name that is displayed on Analytics Workbench. Enter a name that describes your business.
Primary key	You must select one or more primary keys for the properties when you add an object for the first time. After the configuration is complete, you cannot modify or delete the primary keys. If you add a node in Analytics Workbench, you must enter the physical table column mapped by the primary key properties. For example, if the ID card number is the primary key, you need to enter the ID card number when you add an ID card node in Analytics Workbench.

Parameter	Description
Show in Graph	If you select this parameter, **Property Name** of this property will be displayed in **Graph** in Analytics Workbench together with the object node. Otherwise, Property Name will not be displayed. For example, if you select the ID card number, this number will be displayed in **Graph** together with the ID card object. If you select the name property at the same time, the name and the ID card number will be displayed together with the ID card object. If you select this parameter for a specific property, a bubble icon appears next to the option. You can click the bubble icon to specify whether to show the **Property Name** in **Graph**. For example, if you select this parameter for the ID card number, and choose to display **Property Name** in **Graph**, the ID card property displayed in **Graph** will be **ID card number: xxxxxx**.
Show in Properties	If you select this parameter for a property, the property will be displayed on the Details tab and the Property tab in the right-side pane of the Graph page in Analytics Workbench. Otherwise, the property will not be displayed.
Conditional Query:	If you select this parameter for a property, you can query the object based on this property in **Target Object** when you perform an analysis on the **Graph** page.
Show in Statistics	If you select this parameter for a property, the property is displayed on the **Statistics** tab in the right-side pane of the Graph page in Analytics Workbench. If you do not select this parameter, the property is not displayed in Analytics Workbench.
Available	If you select this parameter for a property, the property takes effect and is displayed in Analytics Workbench. This parameter must be selected for primary key properties. If any of the following parameters has been selected for the property: **Primary Key**, **Show in Graph**, **Show in Properties**, **Conditional Query** and **Show in Statistics**, the **Available** parameter will be automatically selected for a property. The **Available** parameter will be automatically cleared if you clear all the preceding parameters.
Display Type	After you configure this parameter, the property is displayed on the Details tab and the Property tab in the right-side pane of the Graph page in Analytics Workbench based on the selected type. **Note** To display a property in the format of **Dictionary**, you need to configure a dictionary first.
Query Type	The data type that is supported in the query condition of a property. If you select **Dictionary** for **Display Type**, you must select **Dictionary Option** for **Query Type**.

Parameter	Description
Security Level	The security level for a property. A user with a security level lower than the value of this parameter cannot view the property.
Search Item Configuration Default Query Condition Settings Authorization Code Derived Field	Click the icon in the Actions column corresponding to a property. Specify the following parameters: ○ Search Item Configuration: Search items are displayed in the drop-down list only after they have been configured in Configure a search item. ○ Default Query Condition Settings: the default condition used for an object query. If other properties are used as conditions for a query, this condition is also used by default. ○ Authorization Code: After the authorization code function is enabled, only the authorized users can access this property. ○ Derived Property: After a property is set as a derived property, it is automatically generated based on other properties. Configure the method in which the field is generated as needed.
Delete	If a property is no longer used, you can click the icon to delete this property.
Sort order	The Move Up and Move Down arrows are used to adjust the order of properties that are displayed in Analytics Workbench.

5. (Optional) If you need to add multiple properties, you can refer to the preceding steps to add more properties.
6. (Optional) Specify parameters in **Advanced**.

   Parameters in Advanced describes the parameters in **Advanced**.

Parameters in Advanced

Parameter	Description
Add Image To	The avatar of the object that is displayed in **Graph**. Select a property of the object, and then specify the URL of the image and the suffix of the image. **Add Image To** allows you to specify a combination of the prefix, the property, and the suffix. The prefix is the URL of the image, and the suffix is the image format.
Logical Relation of Authorized Properties	The logical relationship between the authorization codes of properties in each record: ○ **AND**: The current record is visible to the users who meet all authorization code conditions of the properties in this record. ○ **OR**: The current record is visible to the users who meet any one authorization code condition of the properties in this record.

7. Click **Save**.

8.2.6. Configure link properties and business parameters

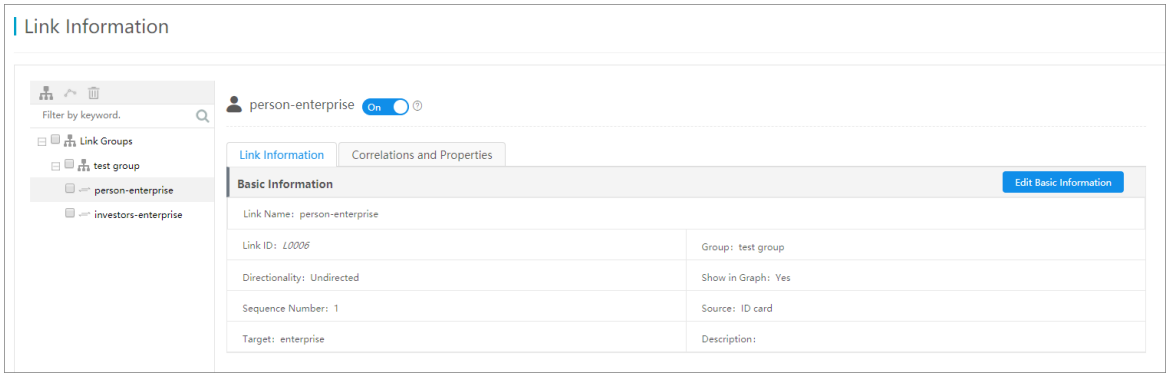
After a first-degree link is added, you need to configure the properties and business parameters of the link based on your business requirements, so that you can view and apply this link in Analytics Workbench. This topic describes how to configure the properties and business parameters of a first-degree link.

Prerequisites

A first-degree link is created. For more information, see Create a first-degree link.

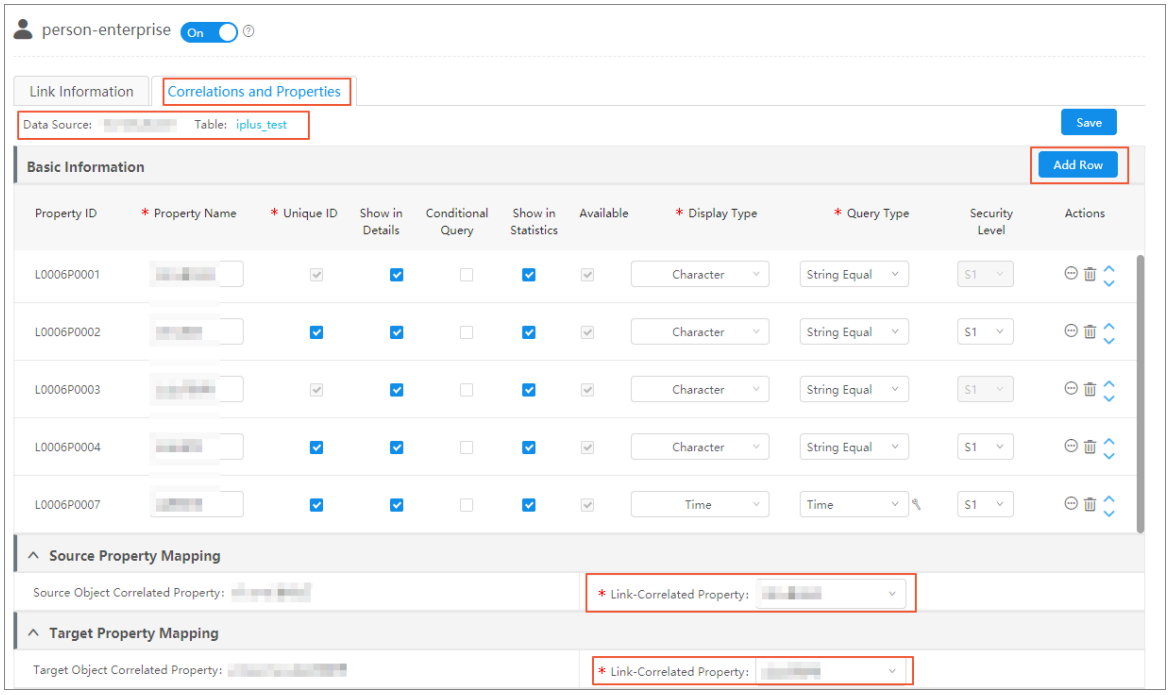
Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. In the left-side navigation pane, click the link group that contains the first-degree link you want to configure, and then click the target link.



4. On the right side of the page, click the **Correlations and Properties** tab.

   On the **Correlations and Properties** tab, if the link is mapped to a data table in the data source, **Data Source** and **Table** are displayed in the property information above the Basic Information section. You can click the table name to redirect to the table page.



5. Configure the parameters in the **Basic Information** section.

   If you need to add a property for a link in the **Basic Information** section, click **Add Row**.

> **Note**
>
> **Link-Correlated Property** in the **Source Property Mapping** section and **Link-Correlated Property** in the **Target Property Mapping** section must be different and both of them must be configured. Therefore, a link must have two or more properties.

Parameters in the Basic Information section describes the parameters in the **Basic Information** section.

Parameters in the Basic Information section

Parameter	Description
Property ID	The ID of a property. It is automatically generated.
Property Name	The name of a property. If you select Available for a property, the name of the property is displayed in Analytics Workbench.
Unique ID	Allows you to define the logical primary key of a data table.
Show in Details	If you select this parameter for a property, the property is displayed on the **Details** tab in Analytics Workbench.
Conditional Query	If you select this parameter for a property, you can query a link based on the link type specified in **Link Type** when you perform analysis on the **Graph** page in Analytics Workbench.

Parameter	Description
Statistics	If you select this parameter for a property, the property is displayed on the **Statistics** tab in the right-side pane of the Graph page in Analytics Workbench. If you do not select this parameter, the property is not displayed in Analytics Workbench.
Available	If you select this parameter for a property, the property takes effect and is displayed in Analytics Workbench. If one of the following parameters is selected for the property, the **Available** option is automatically selected for a property: **Unique ID**, **Show in Details**, **Conditional Query**, and **Show in Statistics**. If all of the preceding parameters are cleared for a property, the **Available** parameter is automatically cleared for the property.
Display Type	After you set the display type, the property is displayed on the **Property** tab in the right-side pane of the Graph page in Analytics Workbench based on the selected type. **Note** If you want to display a property in the **Dictionary** format, configure a dictionary first.
Query Type	The data type that is supported in the query condition of a property. If you select **Dictionary** for **Display Type**, you must select **Dictionary Option** for **Query Type**.
Security Level	The security level for a property. A user with a security level lower than the value of this parameter cannot view the property.
Search Item Configuration **Default Query Condition Settings** **Authorization Code** **Derived Field**	Find the property and click the More icon in the Actions column. Configure the following parameters: ○ Search Item Configuration: Search items are displayed in the drop-down list only after they have been configured in Configure a search item. ○ Default Query Condition Settings: the default condition used for a link query. If other properties are used as conditions for a query, this condition is also used by default. ○ Authorization Code: After the authorization code function is enabled, only authorized users can access this property. ○ Derived Property: After a property is configured as a derived property, it can be generated automatically based on other properties. Configure the method in which the column is generated based on your needs.
Delete	If a property is no longer used, you can click the Delete icon to delete this property.

Parameter	Description
Sort order	The Move Up and Move Down arrows are used to adjust the order of properties that are displayed in Analytics Workbench.

6. Set **Link-Correlated Property** in the **Source Property Mapping** and **Target Property Mapping** sections.

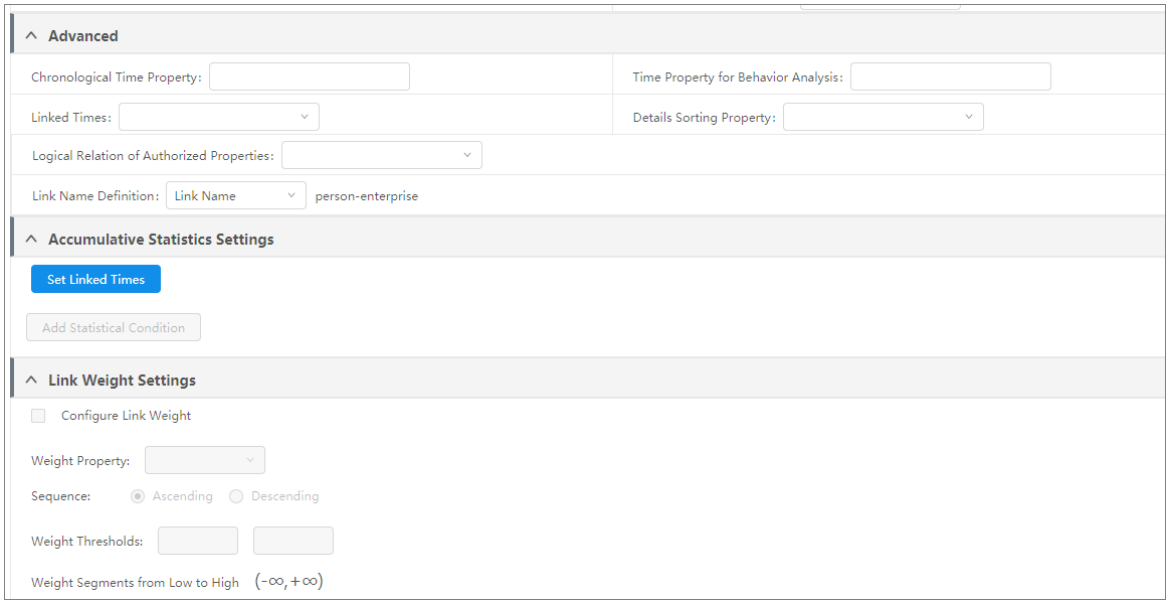
   These configurations are related to Source Object and Target Object of the link.

   Use **Source Property Mapping** as an example. **Source Object Correlated Property** and **Link-Correlated Property** must be mapped to the same column in the same table. The **Source Object Correlated Property** parameter is the primary key property of the source object, which is automatically loaded based on the Source Object parameter. For the **Link-Correlated Property** parameter, you must select the link property that is mapped to the same column in the same table as the Source Object primary key.

   Set **Target Property Mapping** in the same way you set **Source Property Mapping**.

7. (Optional)Configure parameters in the **Advanced**, **Accumulative Statistics Settings**, and **Link Weight Settings** sections based on your requirements.

   For more information about the key parameters, see Key parameters.



Key parameters

Section	Parameter	Description
	Chronological Time Property	The link properties involved in chronological analysis. From the drop-down list, select one or more link properties with the query type of time.
	Time Property for Behavior Analysis	The link properties involved in behavior analysis. From the drop-down list, select one or more link properties with the query type of time.

Section	Parameter	Description
Advanced	Linked Times	The number of the same values that are counted for a specific property in a link. The total number is displayed as the number of link occurrences. The **Linked Times** parameter is used as the default setting to filter link types. Assume that two lines of calls from user A to user C are displayed in the call log. The number of calls from user A to user C is displayed as two in the analysis results.
	Details Sorting Property	The default property by which the returned behavior details are sorted.
Accumulative Statistics Settings	None	Used to perform logical statistics for link properties of which the query type is numeric range. The logical statistics operations include top, equal sign (=), and greater-than-or-equal-to sign (≥). This configuration is suitable for business scenarios where statistics filter is required for link query results. The **Linked Times** parameter is used to filter records in the link query results. You can add statistical conditions to filter the link properties with the query type of numeric range.
Link Weight Settings	None	You can specify a link property with the query type of numeric range and calculate the link weight based on the numeric range specified for the link property.

8. After you modify the parameter configurations, click Save. A message appears, indicating that the modifications are saved.

8.2.7. Configure event properties and business parameters

After you have created an event, you must configure the event properties. Event properties are critical to an event. You can configure event properties, and associate the properties with objects on the Property Information tab.

Prerequisites

You have obtained an account and a password for Graph Analytics and you are granted the required event permissions.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. Click an event in the left-side navigation pane, and click the **Property Information** tab on the right side of the page. The **Property Information** tab appears.

4. Set the required event parameters.

   Click **Add Row** to add a property for a link in **Basic information.**

   The required parameters are displayed in the **Basic Information** and **Set Mappings Between Correlated Objects and Properties** areas. Required event property parameters describes the event parameters.

   > **Note** To save the property information, you must set all the required parameters.

Required event property parameters

Category	Parameter	Description
	Property ID	The ID of a property. It is automatically generated.
	Property Name	The property name that is displayed on Analytics Workbench. We recommend that you enter a name that describes the business type.
	Primary key	Each primary key uniquely identifies an event. A property cannot be deleted after it has been configured as a primary key.
	Show in Graph	**Note** For each event, you must select one or more properties you want to display in the graph. If you select this parameter, the **Property Name** of this property will be displayed in **Graph** in Analytics Workbench together with the object node. Otherwise, the property is not displayed. For example, if this parameter is selected for the ID card property of an event, the ID card number will be displayed in **Graph** with the event. If the name property is also selected, the name and the ID card number will be displayed with the event. If you select this parameter for a specific property, a bubble icon appears next to the option. You can click the bubble icon to specify whether to show the **Property Name** in **Graph**. For example, if you select this parameter for an ID card number, and choose to display the **Property Name** in **Graph**, the ID card property displayed in **Graph** will be **ID card number: xxxxxx**.

Category	Parameter	Description
Basic information	Show in Properties	If you select this parameter for a property, the property will be displayed in the right-side pane on the **Graph** page of Analytics Workbench. To view the property, choose **Properties** > **Event Properties**.
	Element Identifier	The element identifier of a property. Set this parameter based on the actual property.
	Conditional Query	If you select this parameter for a property, the event can be queried based on this property in **Link Type** on the **Graph** page of Analytics Workbench when you perform a relationship analysis.
	Statistics	If you select this parameter for a property, the event will be displayed in the right-side pane on the **Graph** page of Analytics Workbench. To view the property, choose **Statistics** > **Event Distribution**.
	Available	If you select this parameter for a property, the property takes effect and can be displayed on the Graph page of Analytics Workbench. This parameter is selected by default and cannot be changed.
	Display Type	The format in which a property is displayed in the right-side pane of the **Graph** page on Analytics Workbench. Set this parameter as needed.
	Query Type	The data type that is supported in the query condition of a property.
	Security Level	The security level for a property. A user with a security level lower than the value of this parameter cannot view the property.

Category	Parameter	Description
	Search Item Configuration	Click the More icon in the Actions column that corresponds to a property. Configure the following parameters: ○ Search Item Configuration: Search items are displayed in the drop-down list only after they have been configured in Configure a search item. ○ Default Query Condition Settings: The default condition used for a link query. If other properties are used as conditions for a query, this condition is also used by default. ○ Authorization Code: After the authorization code function is enabled, only authorized users can access this property. ○ Derived Property: After a property is set as a derived property, it can be generated automatically based on other properties. Configure the method in which the column is generated based on your needs.
	Default Query Condition Settings	
	Authorization Code	
	Derived Field	
	Delete	If a property is no longer used, you can click the Delete icon to delete this property.
	Sort order	The Move Up and Move Down arrows are used to adjust the order of properties displayed in Analytics Workbench.
Set Mappings Between Correlated Objects and Properties	Add Correlated Object	Adds a mapping between an object and the event. One event must have two or more objects mapped to it. Click **Add Correlated Object** to add a correlated object, and configure the mapping between the event and the primary keys of the object you have added.

5. (Optional)After you have specified the required parameters, you can specify the optional parameters as needed.

   The optional parameters are included in the **Advanced** and **Display Settings** areas. The optional parameters are described in Optional event property parameters.

   > **Note** Location Settings is currently not supported.

Optional event property parameters

Section	Parameter	Description
Advanced	Behavior Property	The properties based on which a behavior analysis is performed.
	Default Details Sorting Property	The property by which the details are sorted.
	Logical Relation of Authorized Properties	The logical relationship between the authorization codes of properties in each record. ○ AND: The current record is visible only to the users who meet all authorization code conditions of the properties in this record. ○ OR: The current record is visible to the users who meet any one authorization code condition of the properties in this record.
Display Settings	Enable Display	Indicates whether to show the event details.
	Group-by Properties	The property based on which events are aggregated. For example, aggregate **Travel Events** into a folder based on the **Train Number** property.

6. After the configurations are complete, click **Save** in the upper-right corner. A success message is displayed after the modifications are saved.

   An event is automatically enabled after its properties are saved.

8.2.8. Log on to Analytics Workbench

Analytics Workbench is a data analysis platform of Graph Analytics. After you configure relevant data in Administration Console of Graph Analytics, you can perform data analysis in Analytics Workbench.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

> **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
>
> - Uppercase or lowercase letters.
> - Digits.
> - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Product** > **Big Data** > **Graph Analytics**.
5. On the Graph Analytics page that appears, specify **Organization** and **Region**, and click **Graph Analytics(I+)** to go to the homepage of Analytics Workbench.

> **Note** You cannot use the root organization for the redirection.

8.2.9. Create an analysis

After you log on to Analytics Workbench, you must create an analysis and add objects as nodes before you perform an analysis.

Prerequisites

- You have obtained an account and a password with the permission to perform graphic operations.
- You have created source tables, objects, links, and events.
- You have obtained data in the tables that have been mapped to the primary keys of the objects you want to analyze. You can query the tables in the database to obtain the data.

Procedure

1. Log on to Analytics Workbench.
2. Click **Create Analysis**. A **Temporary Analysis** tab appears.
3. Click **Add** in the toolbar and then click the blank space, or right-click the blank space and select **Add Node**. Specify parameters in the **Add Node** dialog box that appears.

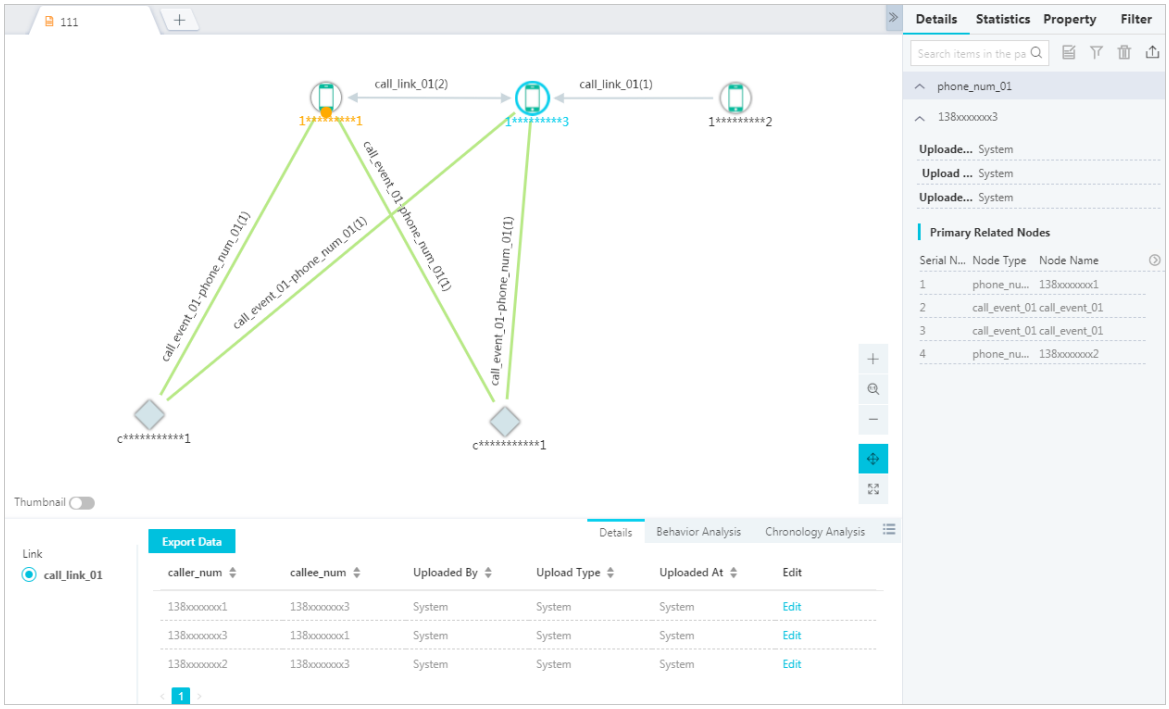
   Parameters to add a node describes the parameters.

Parameters to add a node

Parameter	Description

Parameter	Description
Object type	Displays all created objects. You can select an object as needed. **Note** Graph Analytics allows you to add compound nodes. For example, you can specify two sub-types for object type "person": ID card and passport. The person can be uniquely identified by the ID card or the passport.
Text area	Enter one or more primary key values. Separate multiple primary key values with commas (,).

4. Click **OK**.
5. Right-click a node that has been added, and select **Quick Extension**. The system automatically performs a link analysis based on the configured data sources, objects, links, and events, and displays the analysis results in a graph.



6. Select one or more objects, links, or events. Click **Behavior Chronology** in the lower-right corner to view the information on the **Details**, **Behavior Analysis**, and **Chronology Analysis** tabs.
7. Select one or more objects, links, or events. Click the « icon in the upper-right corner of the right-side pane to view the information on the **Details**, **Statistics**, **Property**, and **Filter** tabs.
8. After the analysis is complete, click **Save** in the upper-right corner. In the **Save Analysis** dialog box, specify **File Name**, select a folder, and then click **OK**. A success message appears after the file has been saved.

   After you have saved the analysis file, if a collaborative analysis is required, you can share

this personal analysis with other members.

9. Click the **Share** icon in the upper-right corner to specify the members you want to share this analysis with.

8.2.10. View analyses

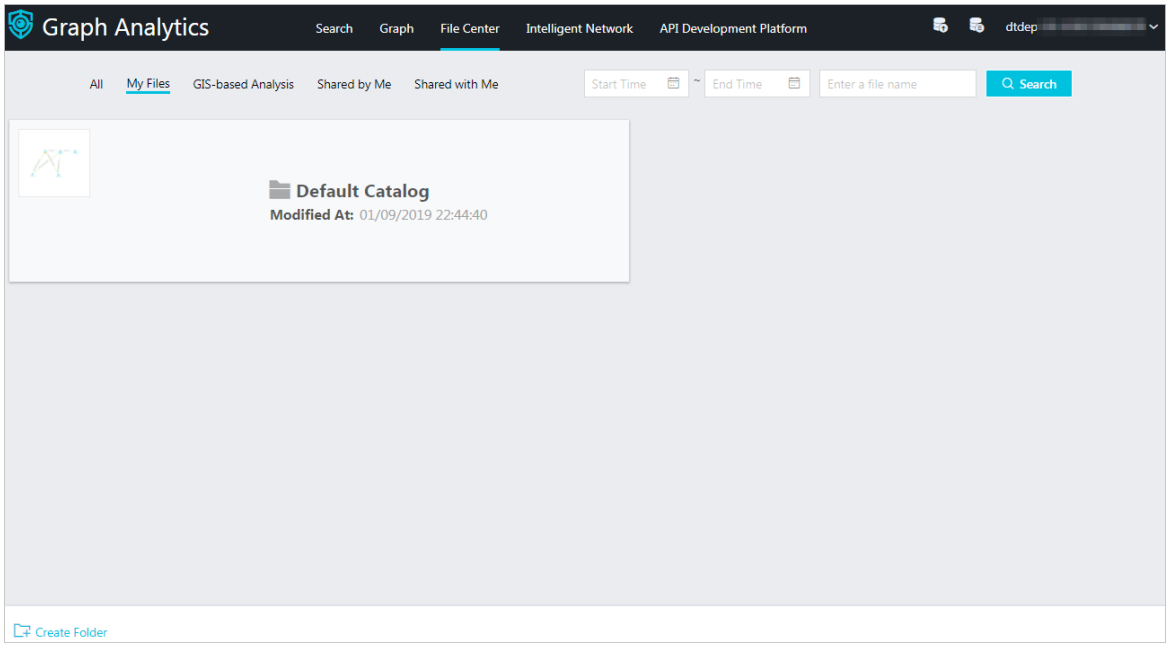
After you close the analysis file or log on to Analytics Workbench again, you can view the existing analyses or shared analyses in **File Center**.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Procedure

1. Log on to Analytics Workbench.
2. Click **File Center** in the top navigation bar. The **File Center** page appears.



3. Click **My Files**, **Shared by Me**, or **Shared with Me** to view the corresponding analysis files.
4. Double-click an analysis to directly open the analysis file on the **Graph** page.

8.3. Source tables

8.3.1. Data sources

8.3.1.1. Create data sources

Before you perform a relationship analysis, you must integrate data that you want to analyze, typically databases, into Graph Analytics. These databases will be used as data sources. In Graph Analytics, every data source is unique and can be added only once.

Prerequisites

- You have an account and password for Graph Analytics and are authorized to perform operations on Administration Console of Analytics Workbench.
- You have obtained the IP address, user name, password, port number, and other information of an accessible data source.

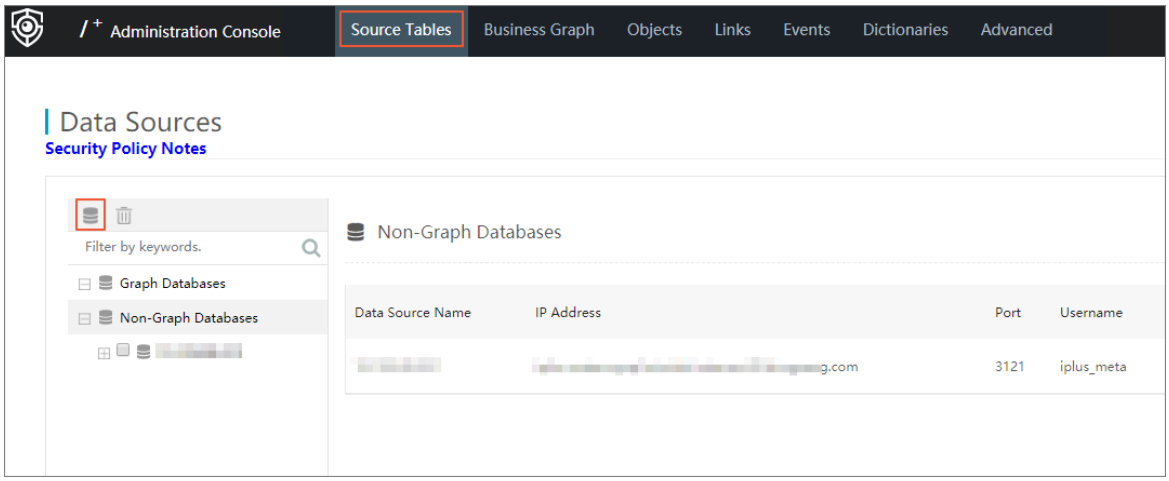
Context

A data source is one of the entries for new objects, links, and events. It is also the only entry for objects, links, and events to map to a data table. Objects, links, and events that are created on the **Object Information**, **Link Information** and **Event Information** pages are logical business objects, links, and events without mappings.

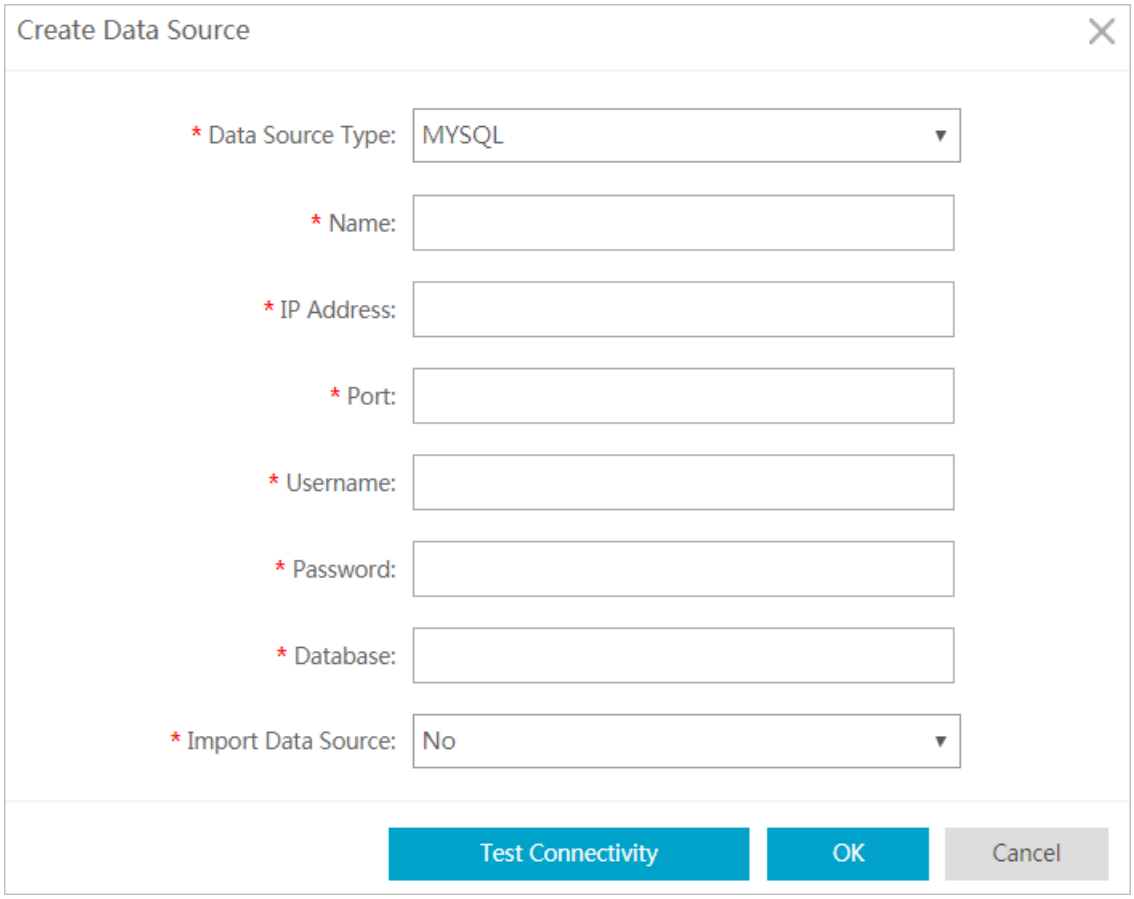
> **Note** Objects, links, and events created for the data source take effect only after you log on again.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Source Tables** to redirect to the **Data Sources** page.



3. On the **Data Sources** page, click the **Create Data Source** icon in the left-side navigation pane. The **Create Data Source** dialog box appears.



4. In the **Create Data Source** dialog box that appears, specify the data source parameters as needed.

Data source parameters

Parameter	Description
Data Source Type	The data source type you want to select. Supported data source types include: **MYSQL**, **ORACLE** , **RDS** , and **GREENPLUM**.
Name	The name of the data source. It can be user-defined.
IP Address	The IP address or domain name of the data source.
Port	The port number of the data source.
Group	The department or group to which the data source belongs.
Username and Password	The username and password that are used to connect to the data source.
Database	The name of the database that functions as the data source.
Import Data Source	If Data Source Type is not set to **ORACLE**, you must specify this parameter. You can import data into Analytics Workbench only after you have specified this parameter.

5. After you have configured the preceding parameters, click **Test Connectivity** to check whether the data source can be connected.

   If the data source is connected properly, a message appears, indicating that the test succeeded. If the data source cannot be connected, check whether the information is correct and the data source itself is functioning properly.

6. After the connectivity test is confirmed as successful, click **OK**.

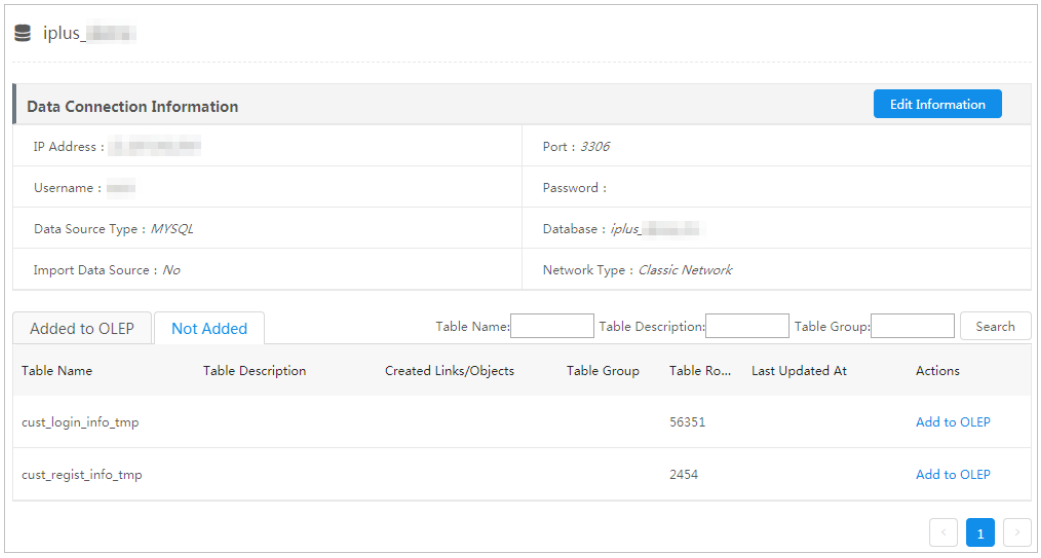
8.3.1.2. View data sources

Graph Analytics allows you to view all data sources in the current environment. This function helps you understand the existing data source information at any time.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar. The **Data Sources** page appears.
3. Select a data source from the left-side navigation pane. The data source details are displayed on the right side of the page, as shown in Data source details.

Data source details



8.3.1.3. Modify a data source

Graph Analytics allows you to modify all data sources in the current environment. This function helps you adjust basic information about data sources at any time if required.

Prerequisites

To modify a data source, you must have the user password that is used to connect to the data source.

Context

You can modify all parameters of a data source except Data Source Type.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar. The **Data Sources** page appears.
3. Click a data source in the left-side navigation pane, and click **Edit Information** in the Data Connection Information section on the right side of the page. Alternatively, click the parent directory of the target data source in the left-side navigation pane, and click **Modify** next to the target data source in the data source list on the right side of the page.
4. In the dialog box that appears, modify the data source parameters as needed, and enter the user **password**.
5. Click **Test Connectivity** to verify the modifications.
6. If the test is successful, click **OK**.

8.3.1.4. Delete a data source

You can delete a data source if it is no longer used.

Prerequisites

All tables in the data source are not correlated with any objects, links, or events. If a data table is correlated with an object, link, or event, remove the association first. For more information, see Remove OLEP tables.

Context

Deleted data sources cannot be recovered. Exercise caution when you delete a data source.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar. The **Data Sources** page appears.
3. Select one or more data sources from the left-side navigation pane, and then click the icon in the top navigation bar.
4. In the Delete Data Sources dialog box that appears, click **OK** to delete the specified data sources.

8.3.2. OLEP tables

8.3.2.1. Create OLEP models for tables

After you add a data source, you must create object, link, event, and property (OLEP) models for the tables in the data source as needed. Before you configure OLEP tables, prepare the tables for which you will create OLEP models, the columns of each table, and the business models to be configured. Referenced tables cannot be deleted.

Prerequisites

You have created an accessible data source.

Context

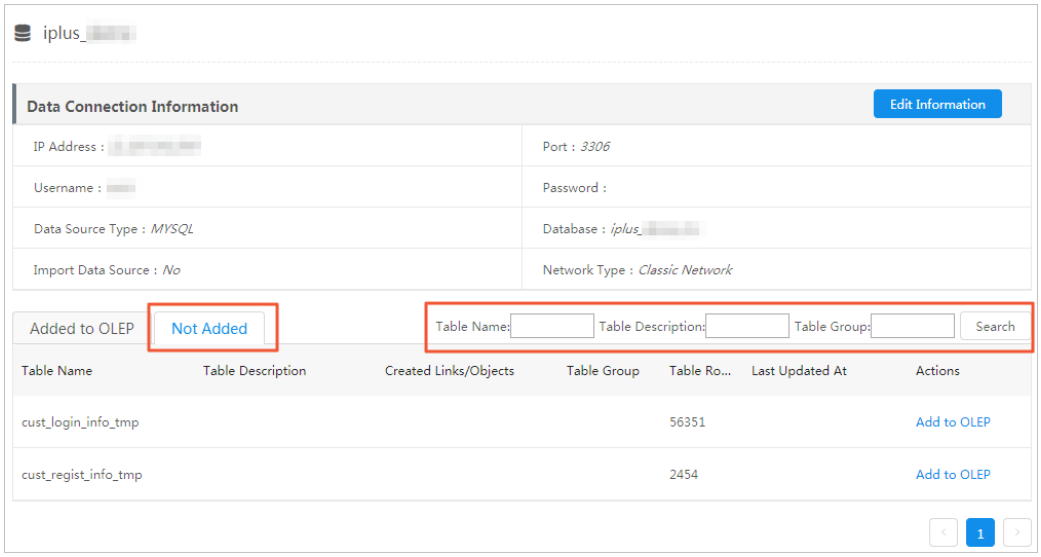
OLEP models include the following three types of mappings: table-to-object mappings, table-to-link mappings, and table-to-event mappings. You can create objects, links, and events when you create OLEP models. Afterward, you can view and configure these objects, links, and events on the **Object Information**, **Link Information**, and **Event Information** pages, respectively. You can configure these items to reflect your business semantics. An OLEP table serves as a source for configuring objects, first-degree links, and events. A table can be mapped to multiple objects, links, and events.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar. The **Data Sources** page appears.
3. Click a data source in the left-side navigation pane. The data source details are displayed on the right side of the page.
4. Click the **Not Added** tab. All tables in the data source that have no OLEP models are displayed.

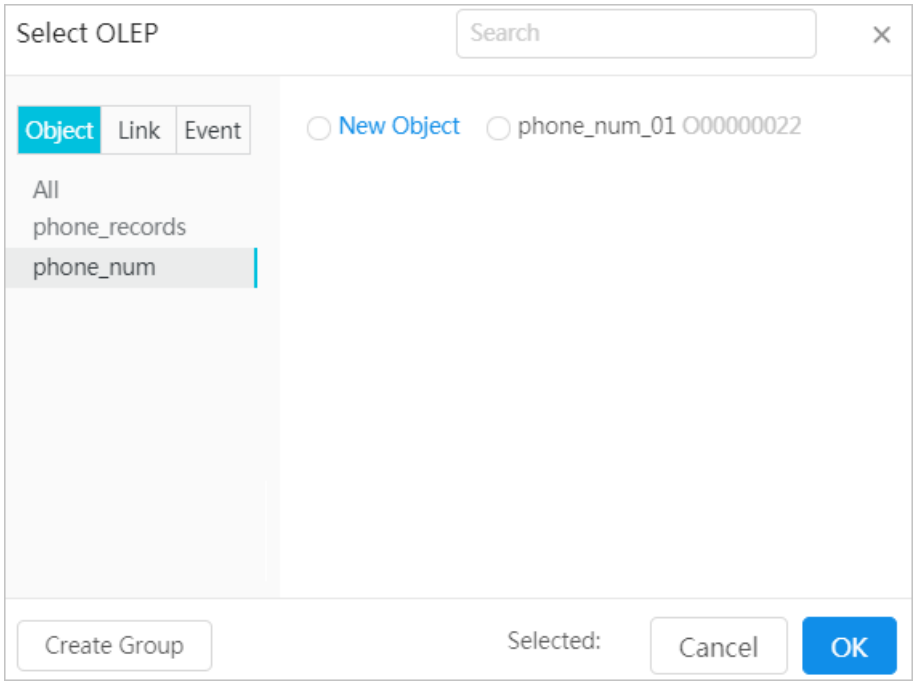
   You can search for a table quickly and accurately by specifying the Table Name, Table Description, or Table Group parameter.

   Tables that have no OLEP models



5. Select a table, and then click **Add to OLEP** in the Actions column. The **Select OLEP** dialog box appears.

   Select OLEP dialog box



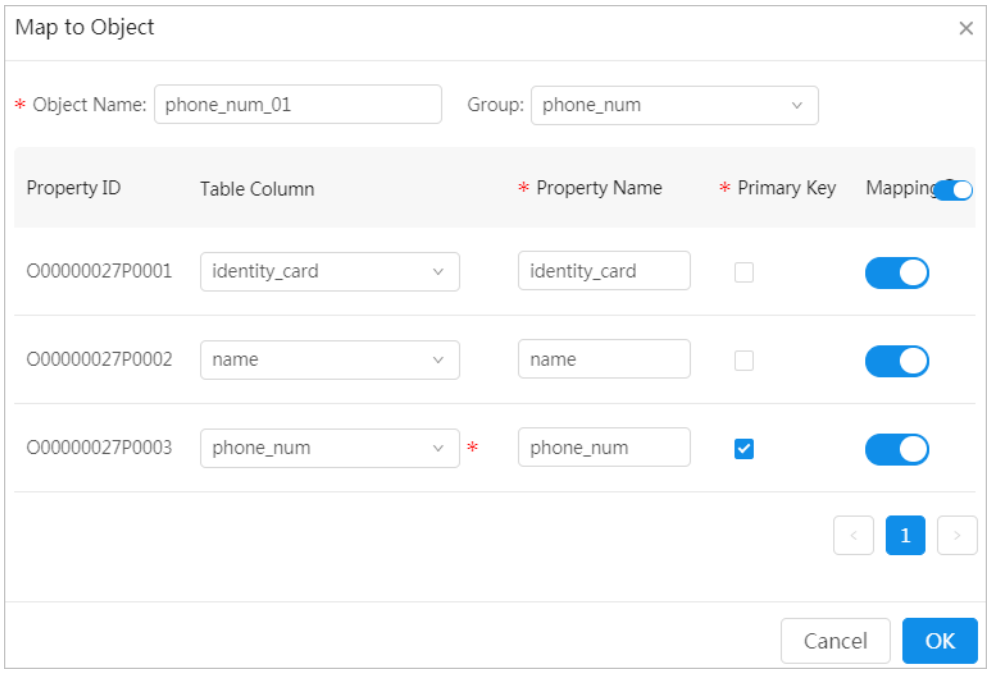
The **Select OLEP** dialog box contains the **Object**, **Link**, and **Event** tabs which are used to create mappings to objects, links, and events, respectively. For more information about how to create a mapping to an object, link, or event, see step 6, step 7, and step 8.

If there are no existing object, link, or event groups that can meet your requirements, click **Create Group** to create a new object, link, or event group.

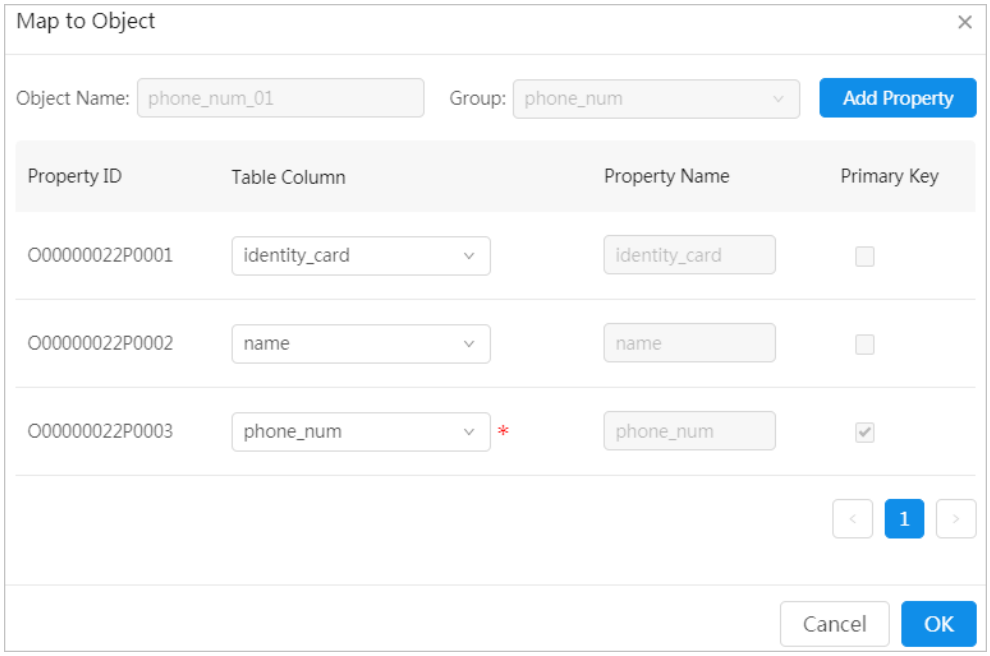
6. Map the table to an object.

i. Select **New Object** or an existing object and then click **OK**. The **Map to Object** dialog box appears.

Map to a newly created object



Map to an existing object



ii. Set the parameters as needed.

- New object: Set parameters based on Parameters used to map the table to a new object.
- Existing object: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns are more than the number of existing properties of the object, you can click **Add Property** to add new properties.

Parameters used to map the table to a new object

Feature name	Description
Object Name	The user-defined object name. It must be unique.
Group	The object group to which the object belongs. All available object groups are displayed in the drop-down list.
Name	The name of an object property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, property names are displayed instead of the actual table columns that are mapped to the properties.
Mapping	Whether to enable the property mapping.
Primary Key	Sets a property as a primary key. Each primary key uniquely identifies an object. You must set one or more properties as primary keys for each object. You must enable **Mapping** for primary keys.

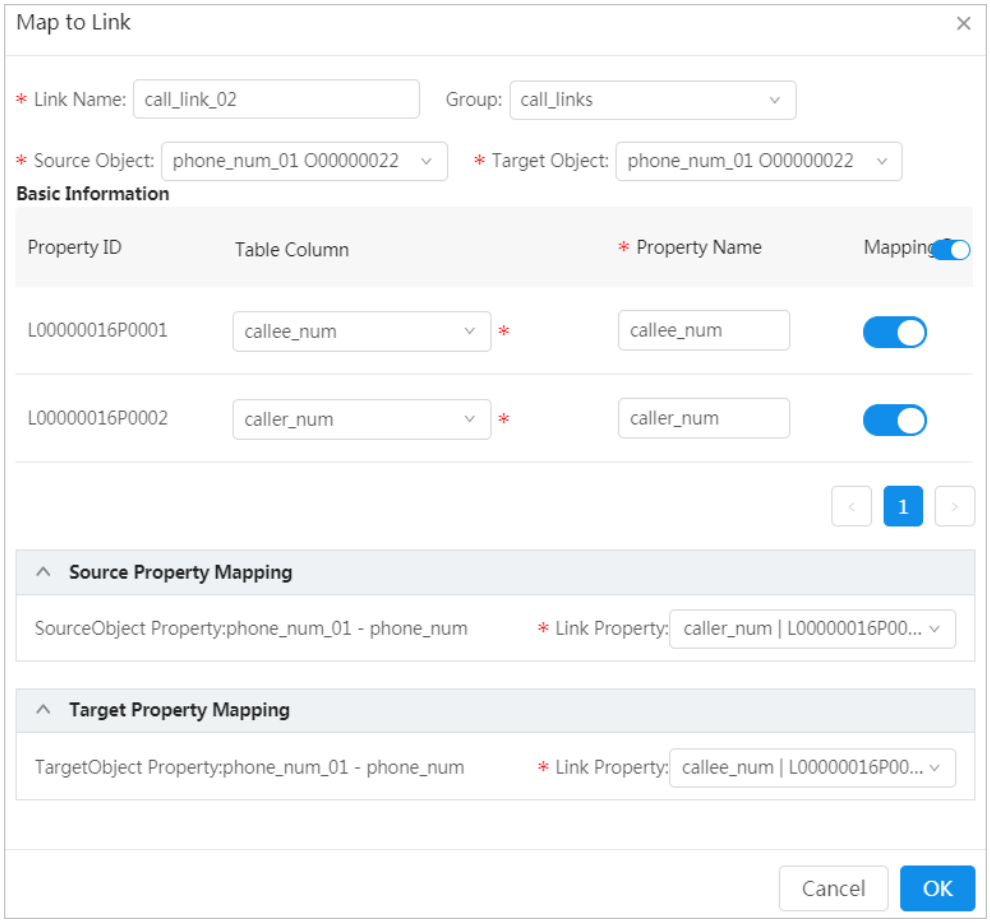
iii. Click **OK**.

7. Map the table to a link.

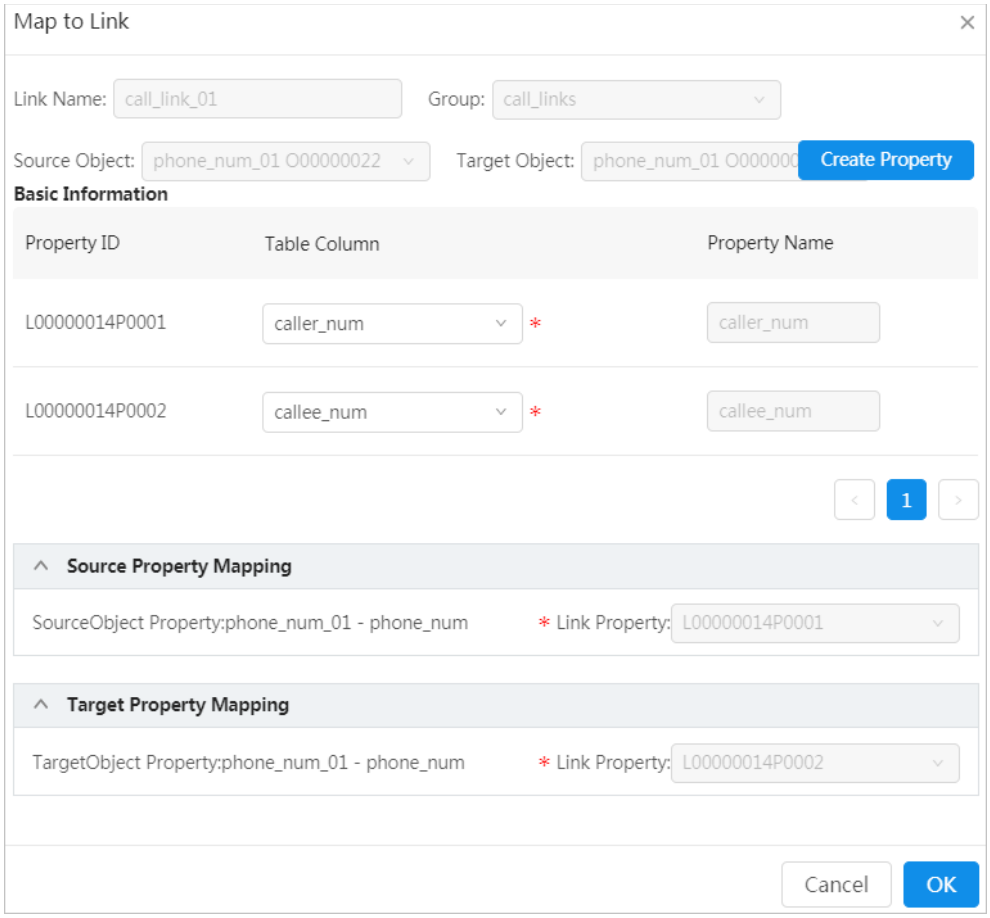
i. Click the **Link** tab. All first-degree links to which the current table has been mapped are displayed.

ii. Select **New Link** or an existing link and then click **OK**. The **Map to Link** dialog box appears.

Map to a newly created link



Map to an existing link


Map to Link
Link Name: call_link_01
Group: call_links
Source Object: phone_num_01 O00000022
Target Object: phone_num_01 O000000
Create Property
Basic Information
Property ID
Table Column
Property Name
L00000014P0001
caller_num
caller_num
L00000014P0002
callee_num
callee_num
1
Source Property Mapping
SourceObject Property:phone_num_01 - phone_num
Link Property: L00000014P0001
Target Property Mapping
TargetObject Property:phone_num_01 - phone_num
Link Property: L00000014P0002
Cancel
OK

iii. Set parameters as needed.

- New link: Set parameters based on Parameters used to map the table to a new link.
- Existing link: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns are more than the number of existing link properties, you can click **Add Property** to add new link properties.

Parameters used to map the table to a new link

Feature name	Description
Link Name	The user-defined link name. It must be unique.
Group	The link group to which the link belongs. All available link groups are displayed in the drop-down list.
Source	The source object of the link. You can select an object from the drop-down list. The **Source Property Mapping** parameter is available only after you set the Source Object parameter.
Target	The target object of the link. You can select an object from the drop-down list. The **Target Property Mapping** parameter is available only after you set the Target Object parameter.
Name	The name of an event property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, the Property Name values are displayed as the table header in **Details** on the **Graph** page.
Mapping	Whether to enable the property mapping.
Link Property in Source Property Mapping	The link property to which a primary key property of the source object is mapped.
Link Property in Target Property Mapping	The link property to which a primary key property of the target object is mapped.

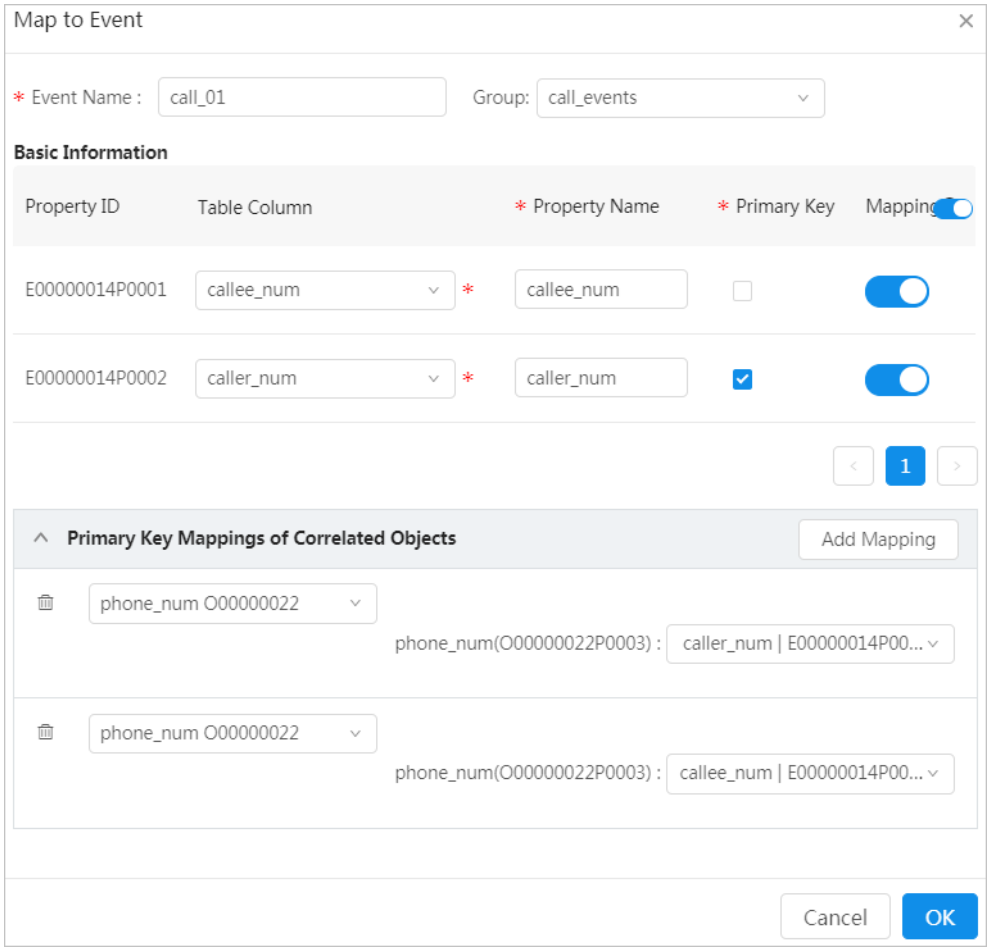
iv. Click **OK**.

8. Map the table to an event.

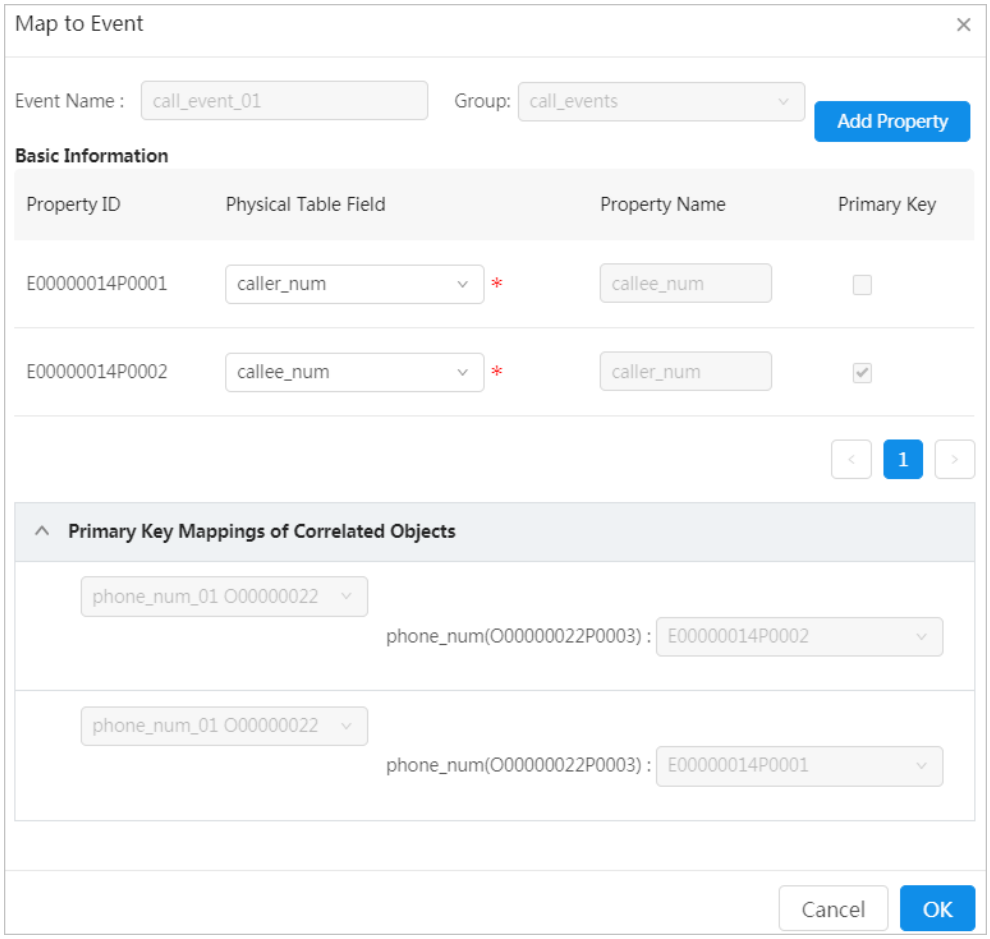
i. Click the **Event** tab. All events to which the current table has been mapped are displayed.

ii. Select **New Event** or an existing event and then click **OK**. The **Map to Event** dialog box appears.

Map to a newly created event



Map to an existing event


Map to Event
Event Name :
call_event_01
Group:
call_events
Add Property
Basic Information
Property ID
Physical Table Field
Property Name
Primary Key
E00000014P0001
caller_num
callee_num
E00000014P0002
callee_num
caller_num
1
Primary Key Mappings of Correlated Objects
phone_num_01 O00000022
phone_num(O00000022P0003) :
E00000014P0002
phone_num_01 O00000022
phone_num(O00000022P0003) :
E00000014P0001
Cancel
OK

iii. Set parameters as needed.

- New event: Set parameters based on Parameters used to map the table to a new event.
- Existing event: Create a one-to-one mapping between each **Table Column** and **Property Name** as needed.

  If the number of table columns are more than the number of existing event properties, you can click **Add Property** to add new event properties.

Parameters used to map the table to a new event

Feature name	Description
Event Definition Name	The user-defined event name. It must be unique.
Group	The event group to which the event belongs. All available event groups are displayed in the drop-down list.
Name	The name of an event property to which a table column is mapped. By default, the property name is the same as the column name. You can also define the property name as needed. On the Analytics Workbench, the Property Name values are displayed as the table header in **Details** on the **Graph** page.
Switch	Whether to enable the property mapping.
Primary Key	Sets a property as a primary key. Each primary key uniquely identifies an event. You must set one or more properties as primary keys for each event. **Switch** must be turned on for the properties that are set as primary keys.
Map Primary Keys to Correlated Objects	Indicates the mappings between the primary keys of correlated objects and the event properties. At least two correlated objects are required. You can click **Add Mapping** to add more necessary mappings between the primary keys of correlated objects and the event properties. You must enable **Mapping** for the event properties to which the primary keys of the correlated objects are mapped.

iv. Click **OK**.

9. After you have created OLEP models for the table, click the **Added to OLEP** tab to check the results.

8.3.2.2. View an OLEP table

After you have created OLEP models for a table, you can view the created OLEP table.

Prerequisites

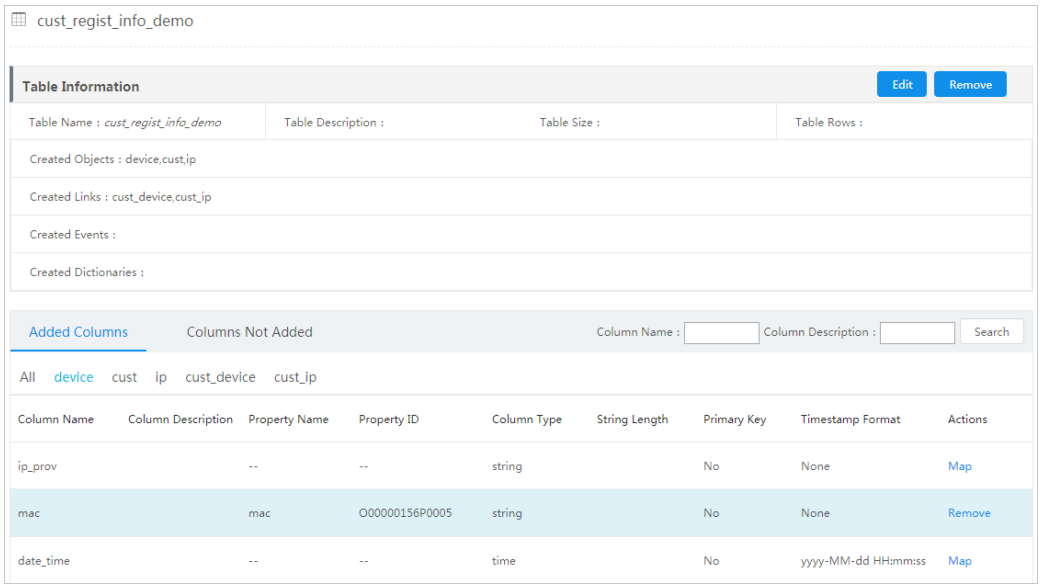
OLEP models have been created for a table.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Select the target table you want to view from the left-side data source list.

   The table information is displayed in the right-side pane, including table basics, dependent objects and links, and table columns, as shown in View tables.

   View tables



8.3.2.3. Edit OLEP tables

After you have created OLEP models for a table, you can still add new object, link, or event mappings to the table. You can also add a description for the table.

Prerequisites

Make sure that OLEP models have been created for a table.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click a table in the left-side navigation pane. The data source details are displayed on the right side of the page.

   Add new object, link, or event mappings to tables.

3. On the **Added to OLEP** tab, select a table, and then click **Add Mapping**. The **Select OLEP** dialog box appears. You can add a new object, link, or event mapping to the table. For more information, see the procedure described in Create OLEP models for tables.

   Modify the table description.

4. You can use one of the following methods to modify the table description.

Method	Operation

Method	Operation
On the **Added to OLEP** page	i. Click the Edit icon  next to the **Table Description** column. You can then modify the **Table Description**. ii. Click **OK** to save the changes. A success message is displayed after the operation is completed.
On the table details page	i. Click a table in the left-side navigation pane. On the right side of the page, click **Edit** in the **Table Information** section. You can then modify the **Table Description**. ii. Click **OK** to save the changes. A success message is displayed after the operation is completed.

8.3.2.4. Remove an OLEP table

If the mapping between a table and an object, link, or event is no longer used, you can remove the mapping.

Prerequisites

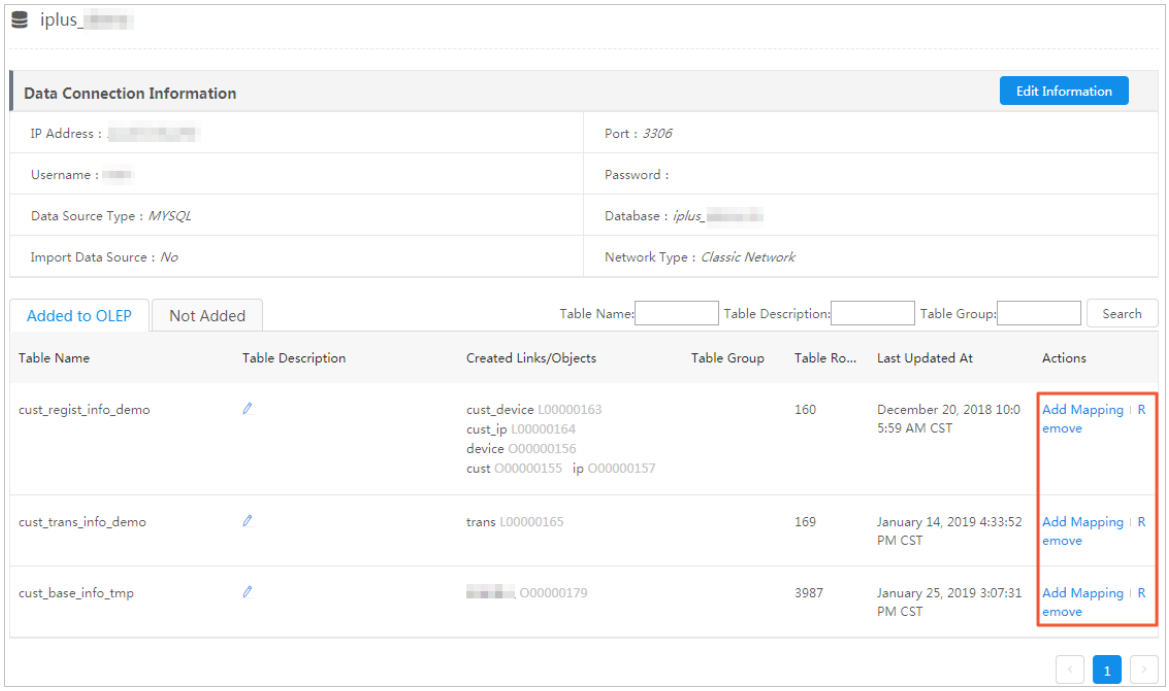
You can access the data source that stores the table.

Context

You only remove the mappings between the tables and objects, links, and events. The objects, links, events, and tables will not be deleted.

Procedure

1. Click the **Source Tables** tab in the top navigation bar. The **Data Sources** page appears.
2. Click a table in the left-side navigation pane. The data source details are displayed in the right-side pane.
3. On the **Added to OLEP** tab, click **Remove** next to a table.



4. In the **Select OLEP** dialog box, select an **Object**, **Link**, or **Event** mapping. You can select only one mapping at a time.
5. Click **OK**.

   If all mappings are removed from a table, the table will be automatically moved from the **Added to OLEP** tab to the **Not Added** tab.

8.3.3. OLEP table columns

8.3.3.1. Add OLEP table columns

If a data table has been mapped to an object, link, or event, and the table still has unoccupied columns (columns that are not correlated with any object, link, or event), you can add these columns to the existing mappings as needed.

Prerequisites

A data table has been mapped to an object, link, or event, but the table still has unoccupied columns.

Context

Before you configure the OLEP table columns, sort out the columns for which you will create OLEP models and data types of the columns, especially the time columns.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar.
3. On the **Data Sources** page, click a data source in the left-side navigation pane, and then click the table to which you want to add the columns.
4. Click the **Columns Not Added** tab in the right-side area. In the **Columns Not Added** tab that

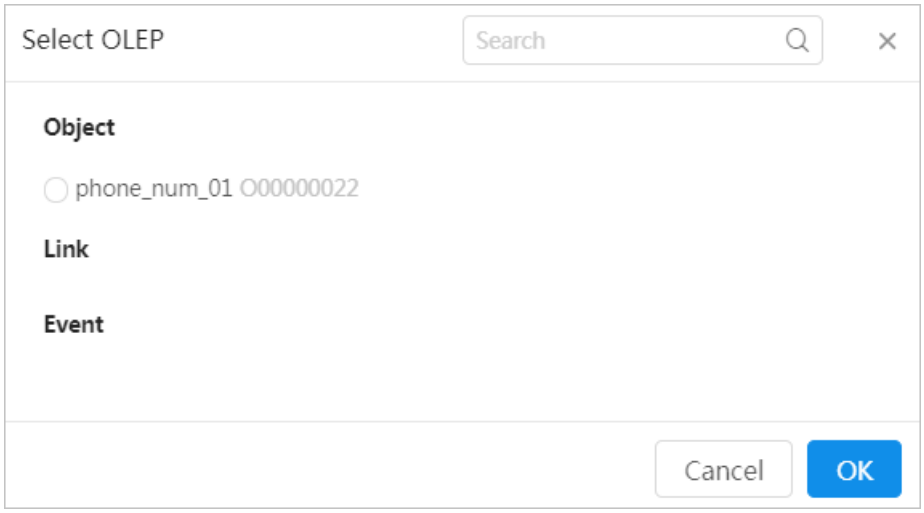
appears, click **Add** in the Actions column.

Columns displayed in the **Columns Not Added** tab are not mapped to any object, link, or event.

5. In the **Select OLEP** dialog box that appears, select the object, link, or event to which the columns map, and then click **OK**.

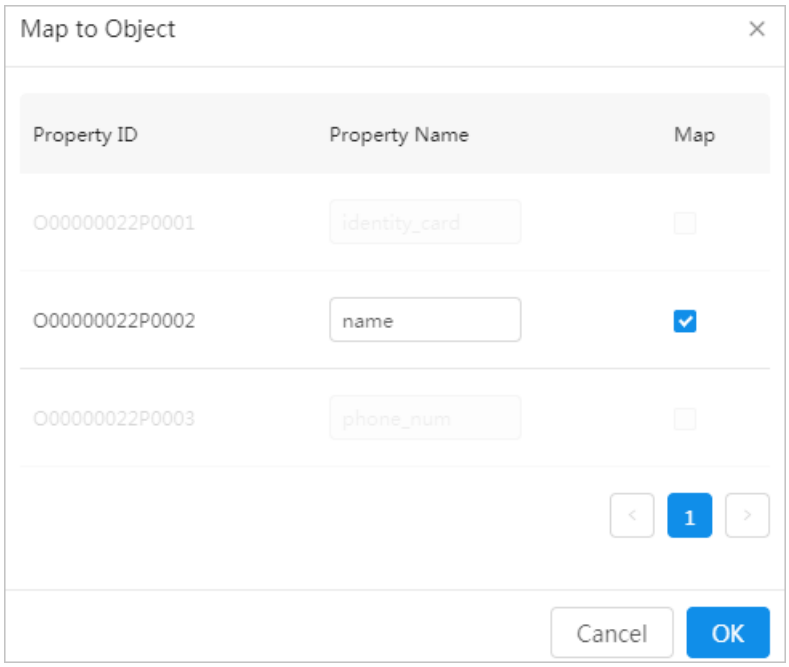
   The **Select OLEP** tab only displays the objects, links, and events that have mapped to the current data table.

   The following example describes how to add columns to an object.



6. In the **Map to Object** dialog box that appears, select the object properties to be mapped to the columns to be added.

   Object properties that have been mapped to the current data table are gray and cannot be operated.



7. After you have configured the preceding parameters, click **OK**.

8.3.3.2. Edit OLEP table columns

After a data table is mapped to objects, links, or events, you can modify the **Column Description** and **Timestamp Format** of added columns.

Prerequisites

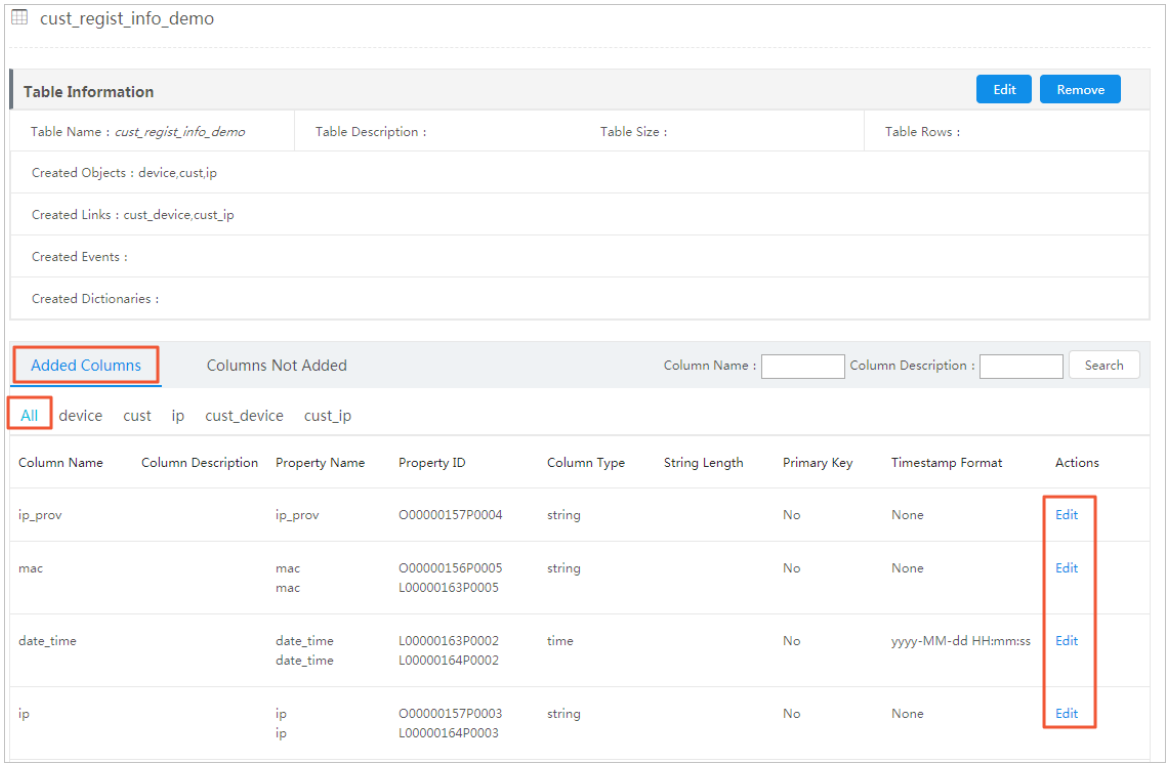
The data table has been mapped to an object, link, or event.

Procedure

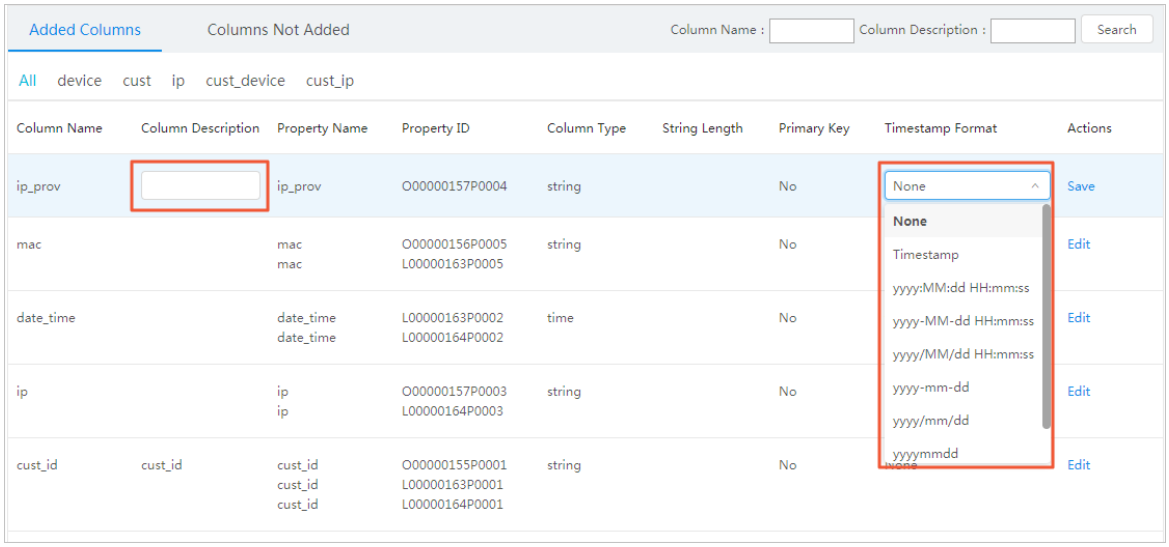
1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar.
3. On the **Data Sources** page, click a data source in the left-side navigation pane, and then click the table that contains the columns you want to modify.

   After you select the table, the details of the table will be displayed in the right-side **Data Sources** pane.
4. Click the **Added Columns** tab.

   By default, the **Added Columns** tab displays all columns that have been added.



5. On the **All** tab, click **Edit** in the Actions column to modify the **Column Description** and **Timestamp Format** of a column.



The **Timestamp** option in the **Timestamp Format** drop-down list refers to the UNIX timestamp.

6. After you have configured the preceding parameters, click **Save**.

8.3.3.3. Remove OLEP table columns

After you have created OLEP models for a table, you can remove unnecessary OLEP mappings of specific columns in the table.

Prerequisites

OLEP table columns are not referenced.

Context

After a table has been mapped to objects, links, or events, you cannot separately remove the following mappings of a column:

- Table-to-object mappings: The mapping between the column and a primary key property of the object cannot be removed.
- Table-to-link mappings: The mapping between the column and a correlated property of the link cannot be removed.
- Table-to-event mappings: The mapping between the column and a primary key property of the event cannot be removed.

To remove these mappings, remove the OLEP mapping from the table and then add OLEP mappings to the table again. For more information, see Remove OLEP tables and Create OLEP models for tables.

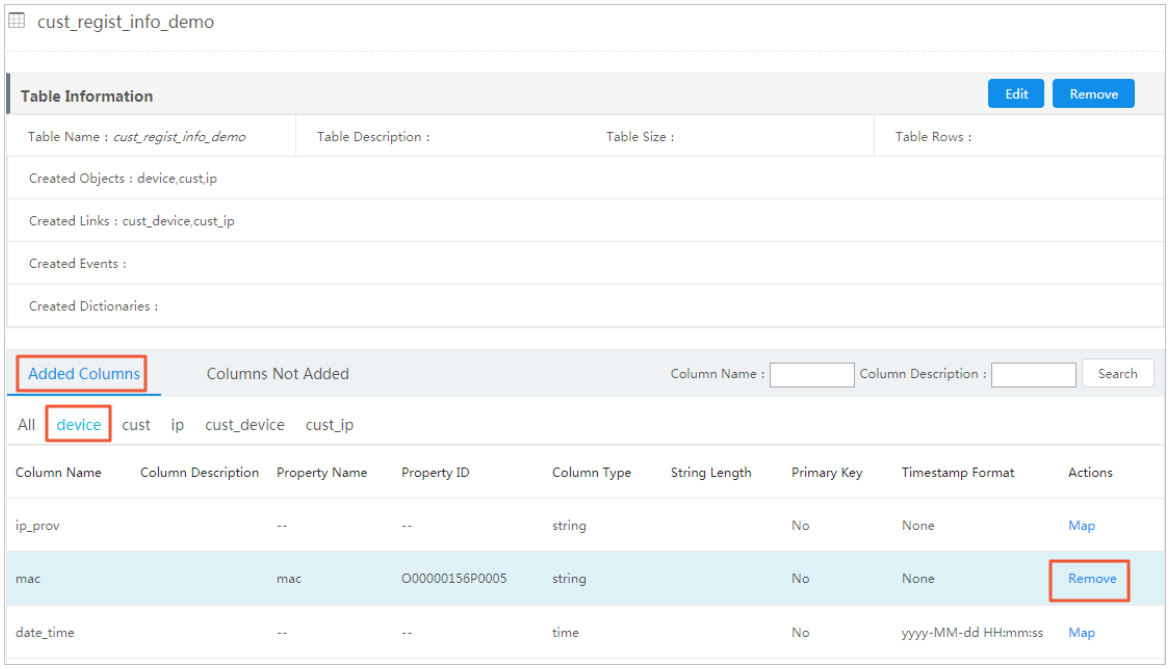
Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Source Tables** in the top navigation bar.
3. On the **Data Sources** page, click a data source in the left-side navigation pane, and then click the table that contains the columns you want to remove.
4. On the right side of the **Data Sources** page, click the **Added Columns** tab.

By default, the **Added Columns** tab displays all columns that have been added.

5. On the **Added Columns** tab, click the tab of an object, link, or event that has been mapped to a data table.

   On the selected object, link, or event page, the table columns that have been mapped are highlighted.



6. Select a column, and then click **Remove** to remove the object, link, or event mapping.

   If all mappings of a column are removed, including object, link, and event mappings, the column will be automatically moved to the **Columns Not Added** tab.

8.4. Dictionaries

8.4.1. Create a dictionary

Before you create a dictionary, sort out the columns to be converted from the system data. If a dictionary has been referenced, it cannot be deleted.

Prerequisites

Before you create a dictionary, complete the following tasks:

- Make sure that you have created a data source. For more information, see Create data sources.
- Make sure that you have configured mappings between tables and objects, links, or events. For more information, see Create OLEP models for tables.
- Map table columns to OLEP. For more information, see Add OLEP table columns.

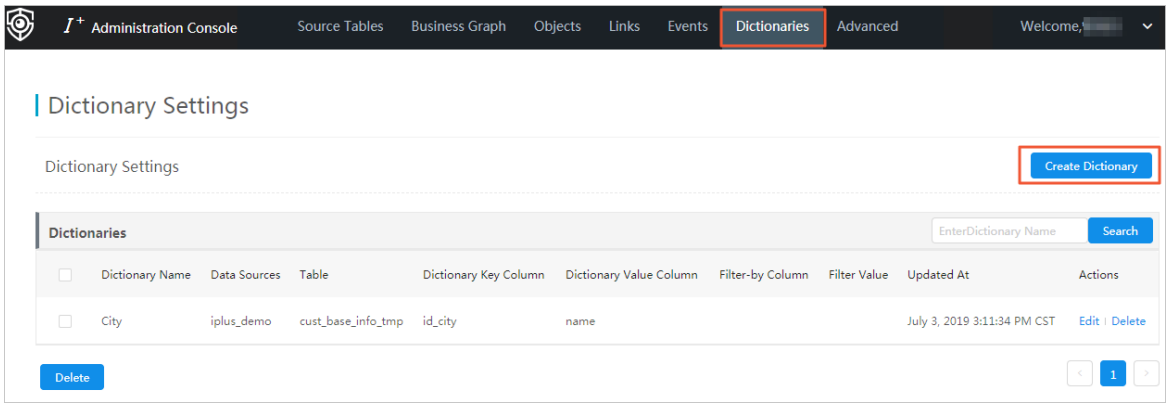
Context

A dictionary is used to map a specific column in the table to a value column and display the value column name in Analytics Workbench instead of the original column name.

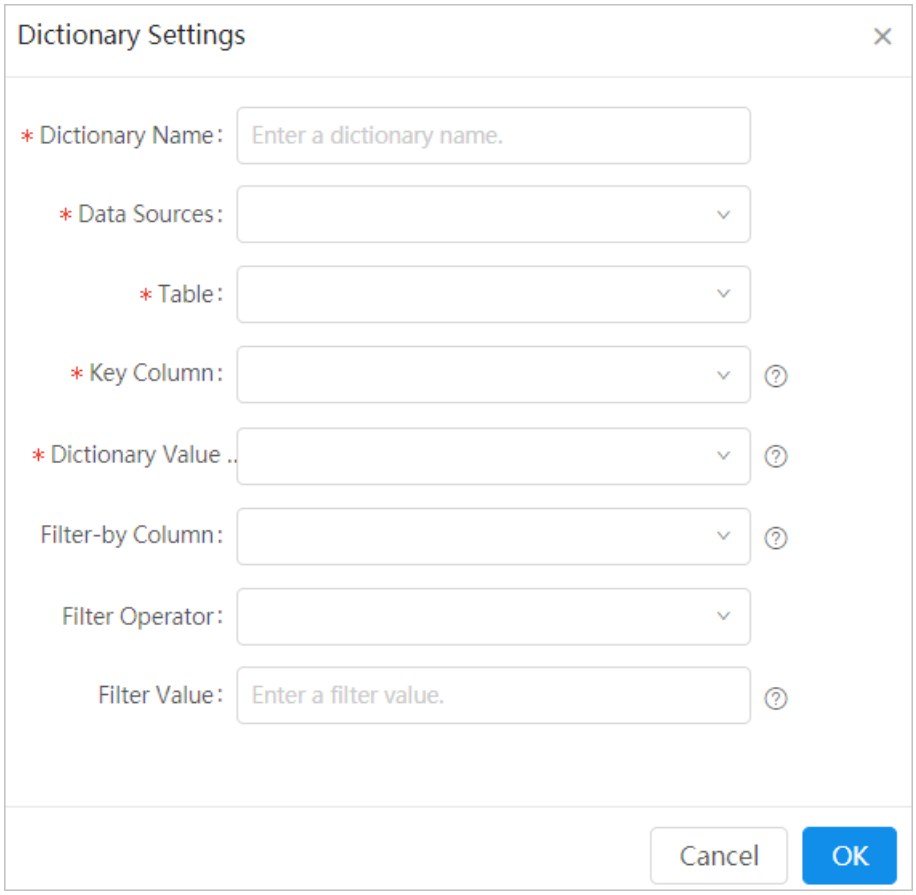
For example, in the **Graph** area of Analytics Workbench, column A (**company code**) is displayed. After column A is escaped to column B (**company name**), the company name corresponding to the code will be displayed in the **Graph** area.

Procedures

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Dictionaries**.



3. On the **Dictionary Settings** page, click **Create Dictionary** in the upper-right corner of the page.



4. In the **Dictionary Settings** dialog box that appears, specify the parameters as needed.

   These parameters are described as follows:

Parameter configurations for adding a dictionary

Parameter	Description
Dictionary Name	The name of the dictionary. The user can customize the name as needed.
Data Sources	The data sources to be referenced.
Table	The table in the data source to be referenced.
Key Column	The column that stores the dictionary code in the selected table.
Dictionary Value Column	The value column corresponding to the converted dictionary.
Filter-by Column	These three parameters are not required and are used to filter dictionary tables based on different conditions. If you specify any of the three parameters, the other two parameters are required.
Filter Operator	
Filter Value	

5. After you have configured the preceding parameters, click **OK**.

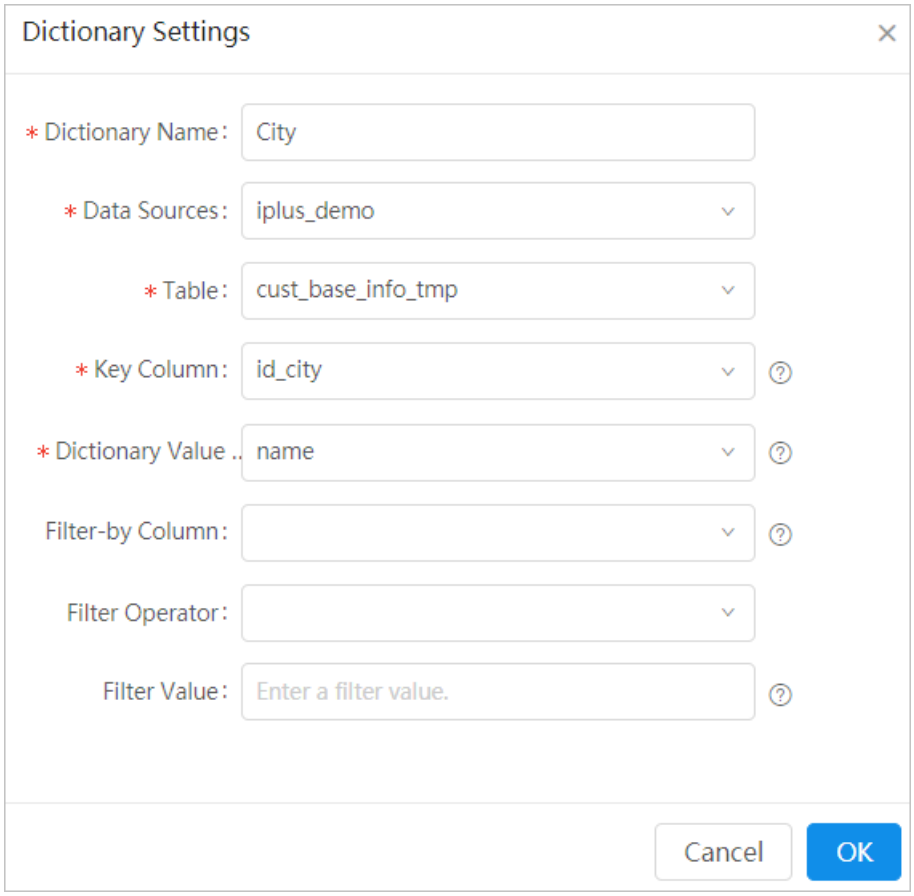
8.4.2. Modify a dictionary

This topic describes how to modify a dictionary in a dictionary list.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Dictionaries**. The Dictionary Settings page appears. All existing dictionaries are displayed.
3. Select a dictionary and then click **Edit**, as shown in Modify a dictionary.

   Modify a dictionary



4. In the dialog box that appears, modify the parameters as needed. For more information about dictionary parameters, see Parameter description.
5. Click **OK**.

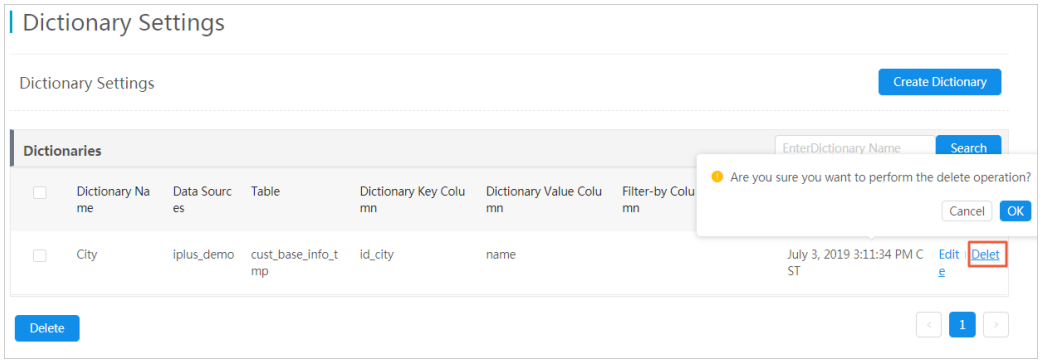
8.4.3. Delete a dictionary

This topic describes how to delete a dictionary. You cannot delete dictionaries that have been referenced.

Delete a dictionary

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Dictionaries**. The Dictionary Settings page appears. All existing dictionaries are displayed.
3. Select a dictionary, and then click **Delete**. A confirm message appears, as shown in Delete a dictionary.

   Delete a dictionary

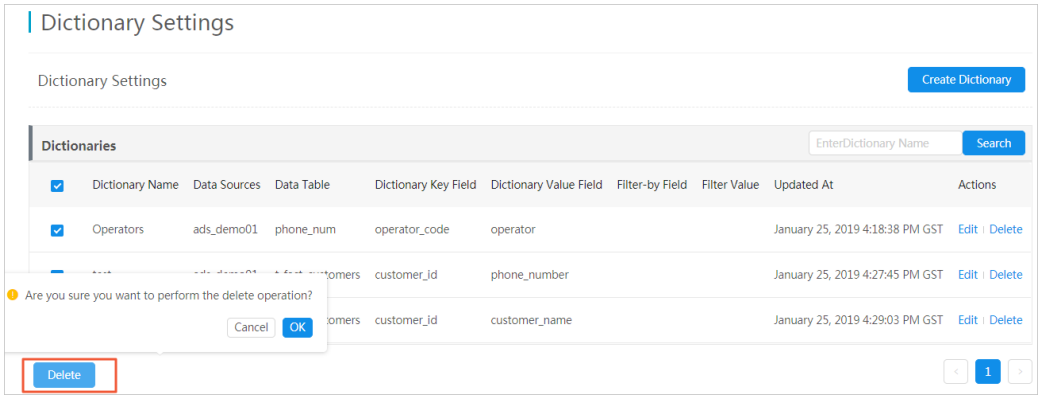


4. Click **OK**.

Delete multiple dictionaries at a time

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Dictionaries**. The Dictionary Settings page appears. All existing dictionaries are displayed.
3. Select one or more dictionaries, and then click **Delete** in the lower-left corner. A confirm message appears, as shown in Delete multiple dictionaries at a time.

   Delete multiple dictionaries at a time



4. Click **OK**.

8.5. Object information

8.5.1. Object groups

8.5.1.1. Create an object group

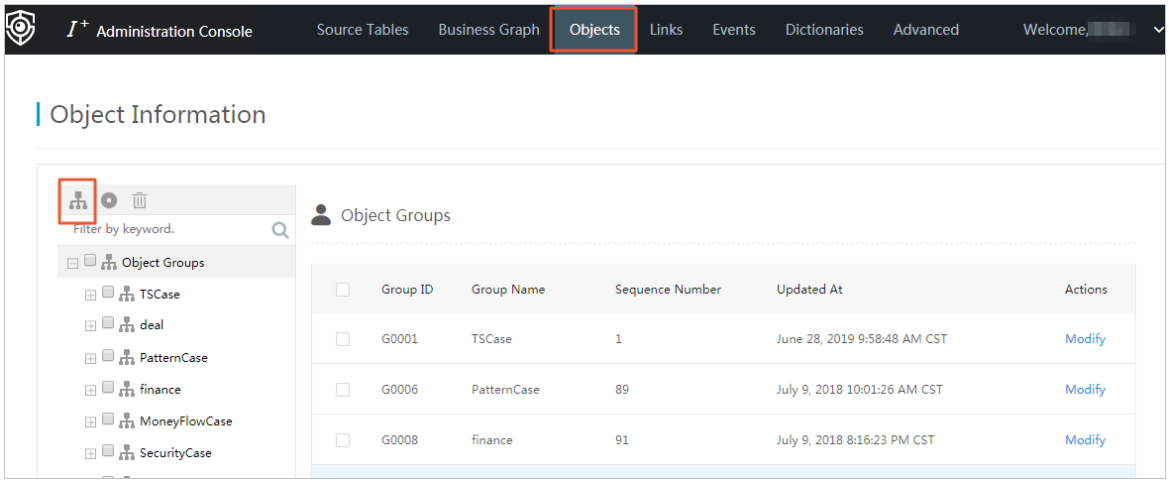
You can use object groups to classify objects, so that you can search for and manage objects with ease. An object must be and can only be included in one object group. You need to create a proper object group before you create an object.

Prerequisites

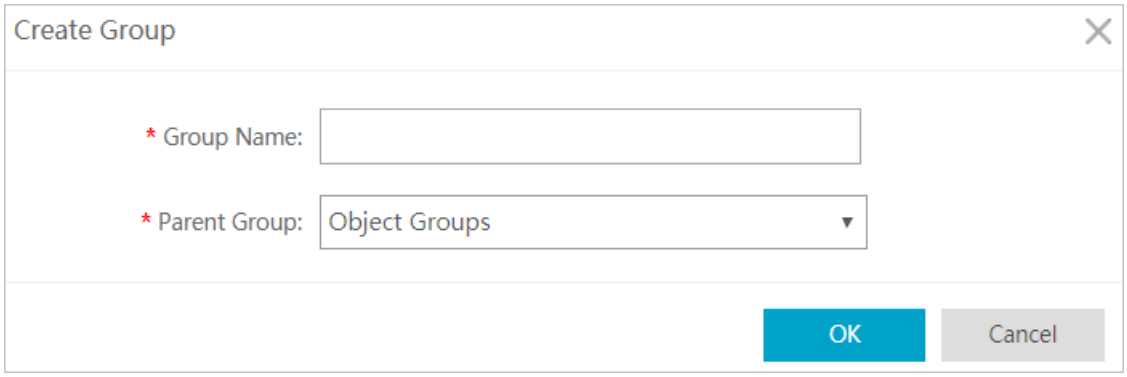
You have an account and password for Graph Analytics and have permissions to perform operations in Administration Console of Graph Analytics.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. Click the **Create Group** icon to create a group.



4. In the **Create Group** dialog box that appears, specify **Group Name** and **Parent Group**.



5. Click **OK**.

8.5.1.2. View object groups and objects

Graph Analytics allows you to view all object groups in the current environment. This function makes it easy for you to understand the existing object groups and the object information under each group at any time.

Prerequisites

You have an account and password for Graph Analytics and have permissions to perform operations in Administration Console of Graph Analytics.

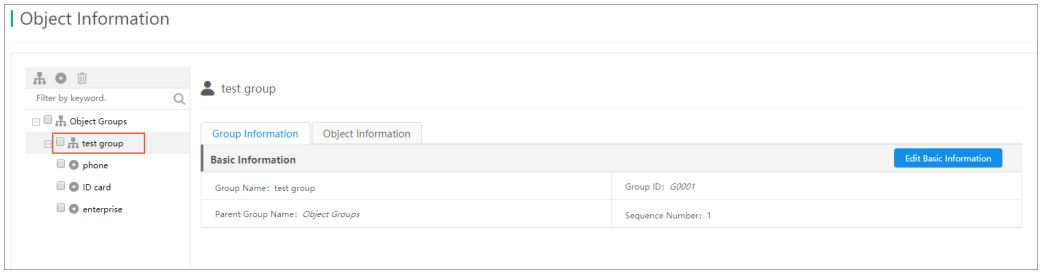
Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane of the **Object Information** page, click an object group. The **Group Information** and **Object Information** tabs of the object group are displayed in the

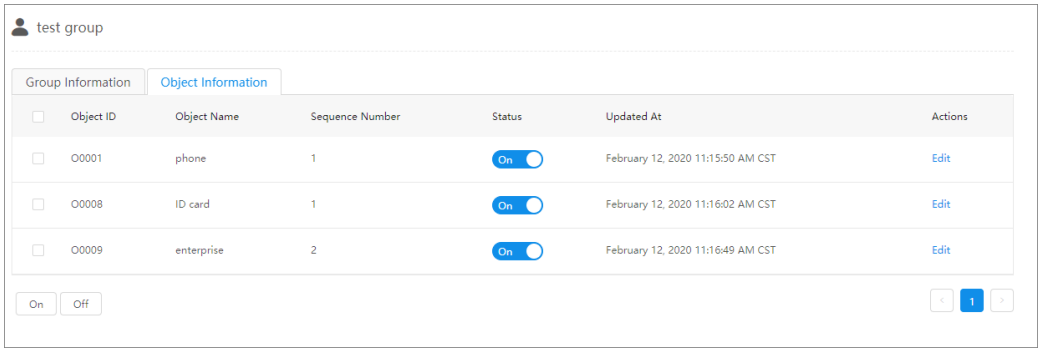
right-side pane.

The left-side navigation pane displays all the object groups in the current environment. You can view the groups one by one.

Group Information



Object Information



4. Click OK.

8.5.1.3. Modify object groups and objects

In Graph Analytics, you can modify the name and sequence number of an object or object group. You can adjust the basic information of an object or object group at any time if required.

Prerequisites

- You have created an object group. For more information about how to create an object group, see Create an object group.
- You have created an object that belongs to this object group. For more information about how to create an object, see Create an object.
- You have deleted the dependency information of an object, including the mapping between the object and a data table if you need to disable this object.

Context

This topic describes the following operations:

- Modify the basic information about an object group
- Modify the basic information about an object
- Disable and enable an object

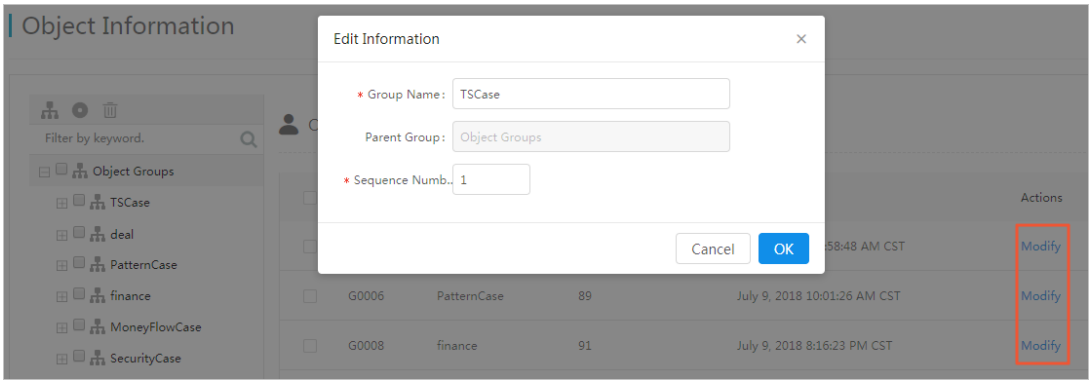
After you have created an object by configuring the object properties, the object is automatically enabled. You can disable an object if it is no longer used for a period of time. You can also enable it again when necessary. You cannot use an object in Analytics Workbench after the object is disabled.

Modify the basic information about an object group

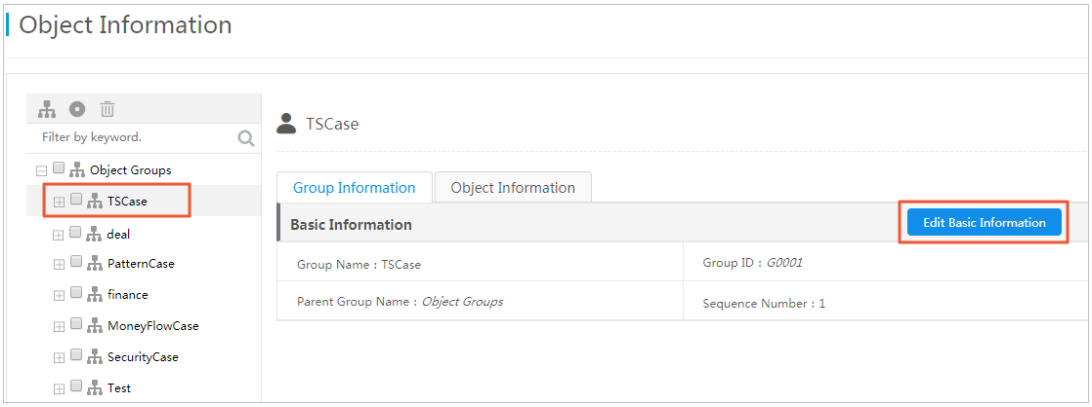
You can modify the basic information about an object group by using one of the following methods:

Method	Procedure
Method 1	1. Log on to Administration Console of Graph Analytics. 2. In the top navigation bar, click **Objects**. 3. In the **Object Groups** area, select an object group and click **Modify** in the Actions column. 4. In the **Edit Information** dialog box that appears, specify **Group Name** and **Sequence Number**, as shown in Edit information. 5. Click **OK**.
Method 2	1. Log on to Administration Console of Graph Analytics. 2. In the top navigation bar, click **Objects**. 3. In the left-side navigation pane, click the object group you want to modify. 4. On the right-side **Group Information** tab, click **Edit Basic Information**. Specify **Group Name** and **Sequence Number**, as shown in Edit information. 5. Click **Save**.

Edit information

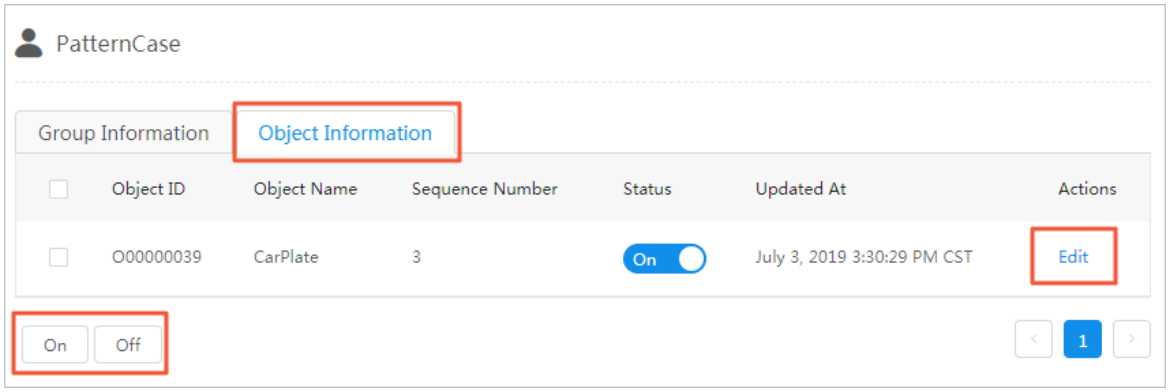


Edit basic information



Modify basic object information

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane, click the object group to which the object belongs, and then click the **Object Information** tab on the right side of the page.
4. On the **Object Information** tab, select an object and then click **Edit** in the Actions column.



5. In the **Edit Information** dialog box that appears, reconfigure **Object Name** and **Sequence Number**.
6. Click **OK**.

Disable and enable an object

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane, click the object group to which the object belongs, and then click the **Object Information** tab on the right side of the page.
4. You can use the following methods to enable or disable an object on the **Object Information** tab:

Method	Procedure

Method	Procedure
Disable an object	Disable an object: Click the On icon next to the object you want to disable. Disable multiple objects: Select the objects you want to disable and then click **Off** in the lower part of the page. After an object is disabled, the **Status** changes from a highlighted On icon to a dimmed Off icon.
Enable an object	Enable an object: Click the Off icon next to the object you want to enable. Enable multiple objects: Select the objects you want to enable and then click **On**. After the object is enabled, the **Status** changes from a dimmed Off icon to a highlighted On icon.

8.5.1.4. Delete object groups and objects

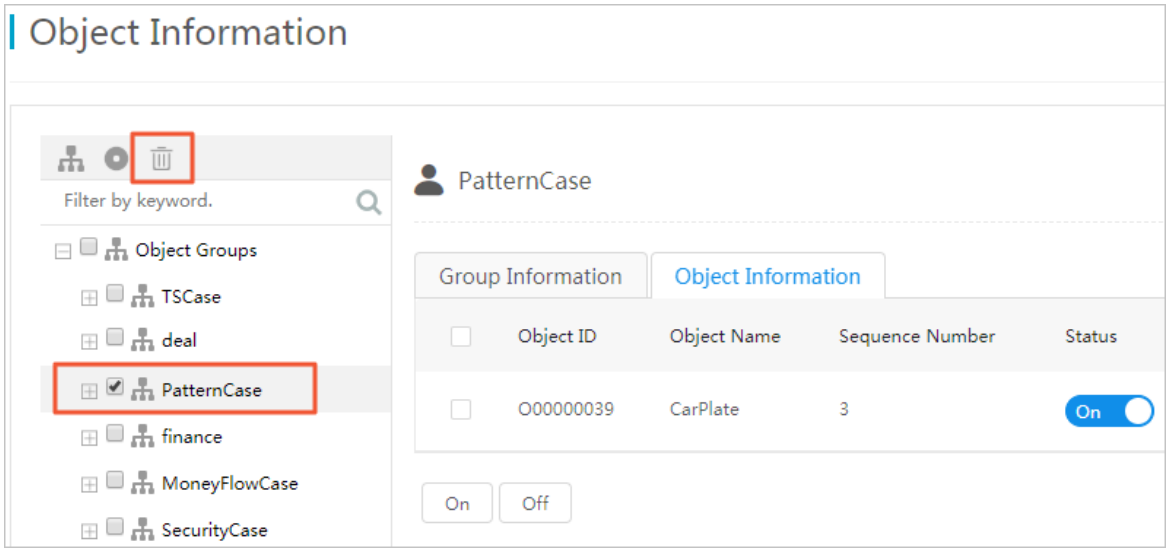
If some objects or object groups are no longer used, you can delete these objects and object groups.

Prerequisites

- The dependency information of an object, for example, the mapping between the object and the physical table, has been deleted before you delete the object.
- All the objects in an object group have been deleted before you delete the object group. Only empty object groups can be deleted.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. (Optional)If there are still objects in the object group, you must delete these objects first.
   i. In the left-side navigation pane, select all objects in the object group you want to delete, and click the icon in the upper-left corner.
   ii. In the **Delete Object Information** dialog box that appears, click **OK** to clear the object group.
4. In the left-side navigation pane, select the object group you want to delete and then click the icon.



5. In the **Delete Object Information** dialog box that appears, click **OK**.

8.5.2. Objects

8.5.2.1. Create an object

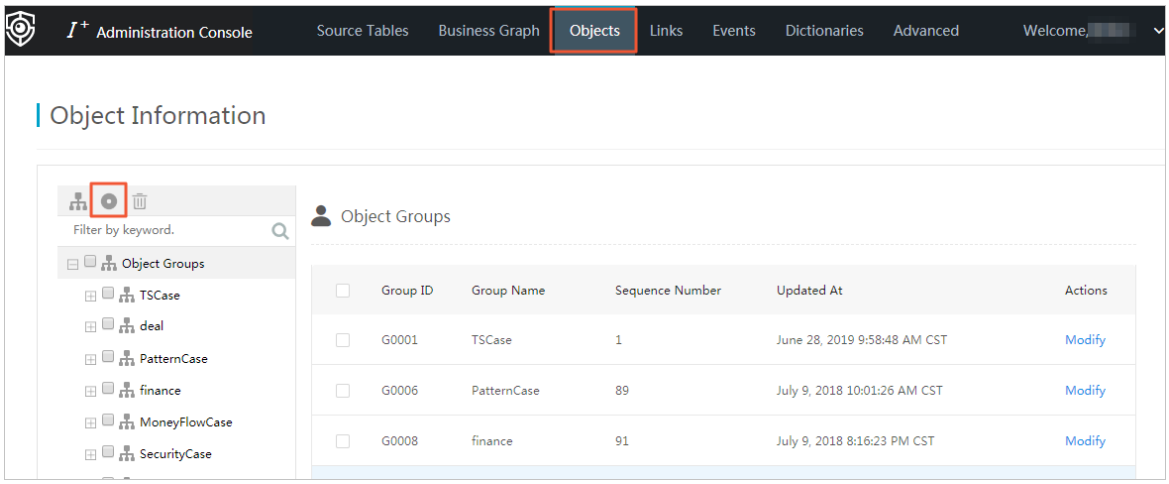
In Graph Analytics, objects are mapped to entities in the real world. Before you perform a relationship analysis on an entity, you must create an object that corresponds to the entity based on the data you have obtained. A complete object contains the basic information, property information, and relevant parameters. This topic describes how to configure the basic information of an object.

Prerequisites

An object group has been created. For more information, see Create an object group.

Procedure

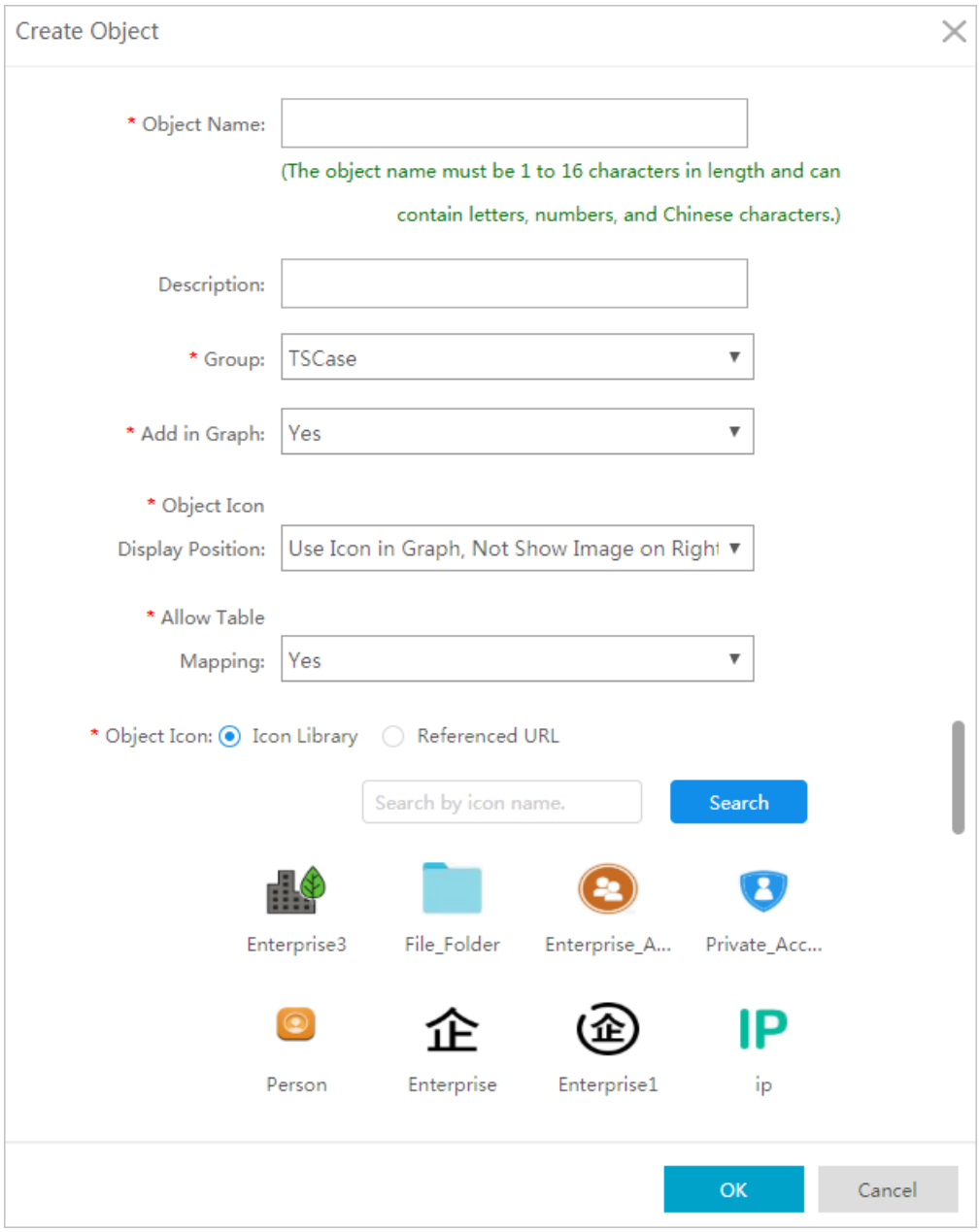
1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.



3. On the **Object Information** page, click the **Create Object** icon.

4. Configure the object information in the **Create Object** dialog box that appears.

   The parameters are described in Parameters of created objects.



Parameters of created objects

Parameter	Description
Object name	The user-defined object name. It must be unique.
Object Description	The object description that helps you understand the object.
Group	By default, the selected object group in the left-side navigation pane is used. You can also select another group as needed.

Parameter	Description
Add in Graph	Indicates whether you are allowed to manually add an object node to the graph page.
Object Icon Display Position	Valid values: ○ Use Icon in Graph, Not Show Image on Right-side Pane ○ Use Icon in Graph, Show Image on Right-side Pane ○ Use Image in Graph, Not Show Image on Right-side Pane ○ Use Image in Graph, Show Image on Right-side Pane
Allow Table Mapping	By default, this parameter is set to Yes.
Object Icon	Allows you to set the icon of the object. You can select an icon in Icon Library or enter a URL to reference an external icon.

5. Click **OK**.

What's next

1. After you have created an object, you must configure the properties and business parameters based on your business requirements. For more information, see Configure object properties and business parameters.
2. After you have configured the properties and business parameters of an object, you must log on to Analytics Workbench again to use the new object.

8.5.2.2. Configure object properties and business parameters

After you add an object, you need to configure the business parameters of the object based on your requirements so that you can view and analyze the object in Analytics Workbench.

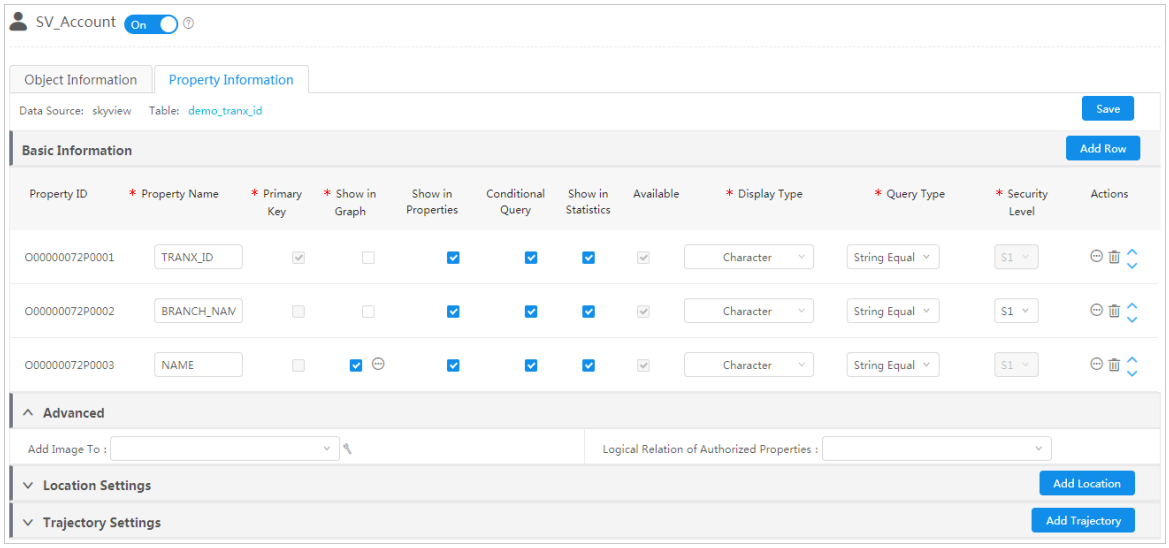
Prerequisites

- You have created a data source. For more information, see Create data sources.
- You have configured mappings between tables and objects, links, or events. For more information, see Create OLEP models for tables.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Objects** on the top of the page.
3. In the left-side navigation pane of the **Object Information** page, click the name of the object you want to configure and then click the **Property Information** tab on the right side.

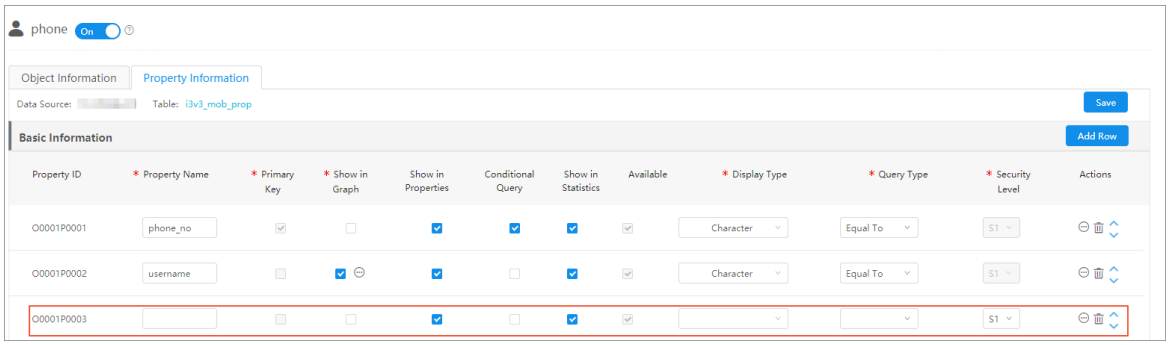
   If this object has been mapped to a physical table in the data source, the **Property Information** tab displays the **Data Source** and **Table** information.



4. Specify parameters in the **Basic Information** section.

   Click **Add Row** to add a property in **Basic information.**

   Parameters in Basic Information describes the parameters in **Basic Information.**



Parameters in Basic Information

Parameter	Description
Property ID	The ID of a property. It is automatically generated.
Property Name	The property name that is displayed on Analytics Workbench. Enter a name that describes your business.
Primary key	You must select one or more primary keys for the properties when you add an object for the first time. After the configuration is complete, you cannot modify or delete the primary keys. If you add a node in Analytics Workbench, you must enter the physical table column mapped by the primary key properties. For example, if the ID card number is the primary key, you need to enter the ID card number when you add an ID card node in Analytics Workbench.

Parameter	Description
Show in Graph	If you select this parameter, **Property Name** of this property will be displayed in **Graph** in Analytics Workbench together with the object node. Otherwise, Property Name will not be displayed. For example, if you select the ID card number, this number will be displayed in **Graph** together with the ID card object. If you select the name property at the same time, the name and the ID card number will be displayed together with the ID card object. If you select this parameter for a specific property, a bubble icon appears next to the option. You can click the bubble icon to specify whether to show the **Property Name** in **Graph**. For example, if you select this parameter for the ID card number, and choose to display **Property Name** in **Graph**, the ID card property displayed in **Graph** will be **ID card number: xxxxxx**.
Show in Properties	If you select this parameter for a property, the property will be displayed on the Details tab and the Property tab in the right-side pane of the Graph page in Analytics Workbench. Otherwise, the property will not be displayed.
Conditional Query:	If you select this parameter for a property, you can query the object based on this property in **Target Object** when you perform an analysis on the **Graph** page.
Show in Statistics	If you select this parameter for a property, the property is displayed on the **Statistics** tab in the right-side pane of the Graph page in Analytics Workbench. If you do not select this parameter, the property is not displayed in Analytics Workbench.
Available	If you select this parameter for a property, the property takes effect and is displayed in Analytics Workbench. This parameter must be selected for primary key properties. If any of the following parameters has been selected for the property: **Primary Key**, **Show in Graph**, **Show in Properties**, **Conditional Query** and **Show in Statistics**, the **Available** parameter will be automatically selected for a property. The **Available** parameter will be automatically cleared if you clear all the preceding parameters.
Display Type	After you configure this parameter, the property is displayed on the Details tab and the Property tab in the right-side pane of the Graph page in Analytics Workbench based on the selected type. **Note** To display a property in the format of **Dictionary**, you need to configure a dictionary first.
Query Type	The data type that is supported in the query condition of a property. If you select **Dictionary** for **Display Type**, you must select **Dictionary Option** for **Query Type**.

Parameter	Description
Security Level	The security level for a property. A user with a security level lower than the value of this parameter cannot view the property.
Search Item Configuration Default Query Condition Settings Authorization Code Derived Field	Click the icon in the Actions column corresponding to a property. Specify the following parameters: ○ Search Item Configuration: Search items are displayed in the drop-down list only after they have been configured in Configure a search item. ○ Default Query Condition Settings: the default condition used for an object query. If other properties are used as conditions for a query, this condition is also used by default. ○ Authorization Code: After the authorization code function is enabled, only the authorized users can access this property. ○ Derived Property: After a property is set as a derived property, it is automatically generated based on other properties. Configure the method in which the field is generated as needed.
Delete	If a property is no longer used, you can click the icon to delete this property.
Sort order	The Move Up and Move Down arrows are used to adjust the order of properties that are displayed in Analytics Workbench.

5. (Optional) If you need to add multiple properties, you can refer to the preceding steps to add more properties.
6. (Optional) Specify parameters in **Advanced**.

   Parameters in Advanced describes the parameters in **Advanced**.

Parameters in Advanced

Parameter	Description
Add Image To	The avatar of the object that is displayed in **Graph**. Select a property of the object, and then specify the URL of the image and the suffix of the image. **Add Image To** allows you to specify a combination of the prefix, the property, and the suffix. The prefix is the URL of the image, and the suffix is the image format.
Logical Relation of Authorized Properties	The logical relationship between the authorization codes of properties in each record: ○ **AND**: The current record is visible to the users who meet all authorization code conditions of the properties in this record. ○ **OR**: The current record is visible to the users who meet any one authorization code condition of the properties in this record.

7. Click Save.

8.5.2.3. Enable and disable an object

After you have created a complete object (configured the object properties), the object is enabled automatically. You can disable an object if it is no longer used for a certain period of time and enable it again when necessary.

Prerequisites

To disable objects, you must first delete the dependency information of these objects, including the mappings between the objects and data tables and the objects referenced by links.

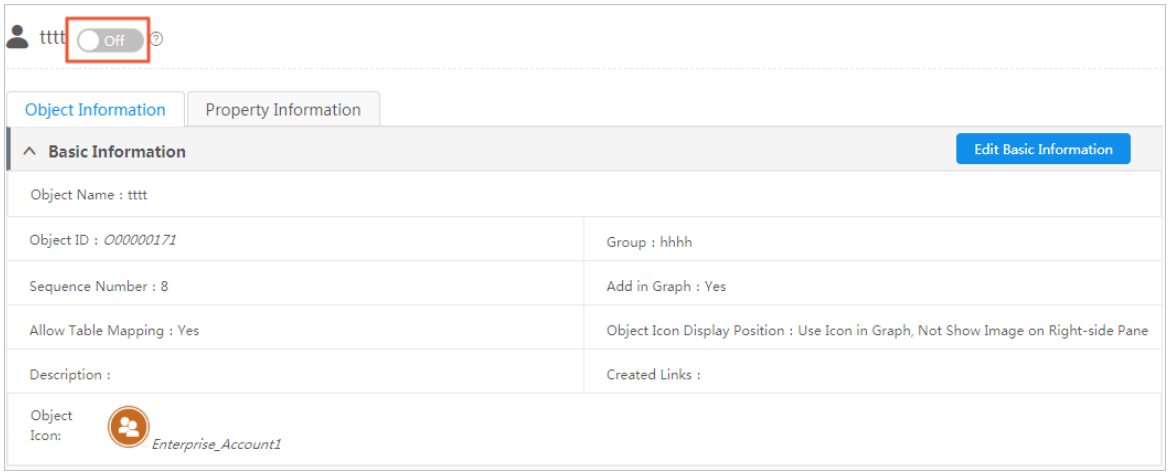
Context

You cannot use an object in Analyitcs Workbench after the object is disabled.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane, click the object group to which the object belongs, and then click the object to be enabled or disabled.

   The detailed information of the object is displayed on the right side of the page.



4. Enable or disable an object as follows.

Operation	Procedure
Disable an object	If the current object is enabled, you can click the On icon to disable the object. After the object is disabled, the **Status** changes from a highlighted On icon to a gray Off icon.

Operation	Procedure
Enable an object	If the current object is disabled, you can click the Off icon to enable the object. After the object is enabled, the **Status** changes from a gray Off icon to a highlighted On icon.

8.5.2.4. Modify an object

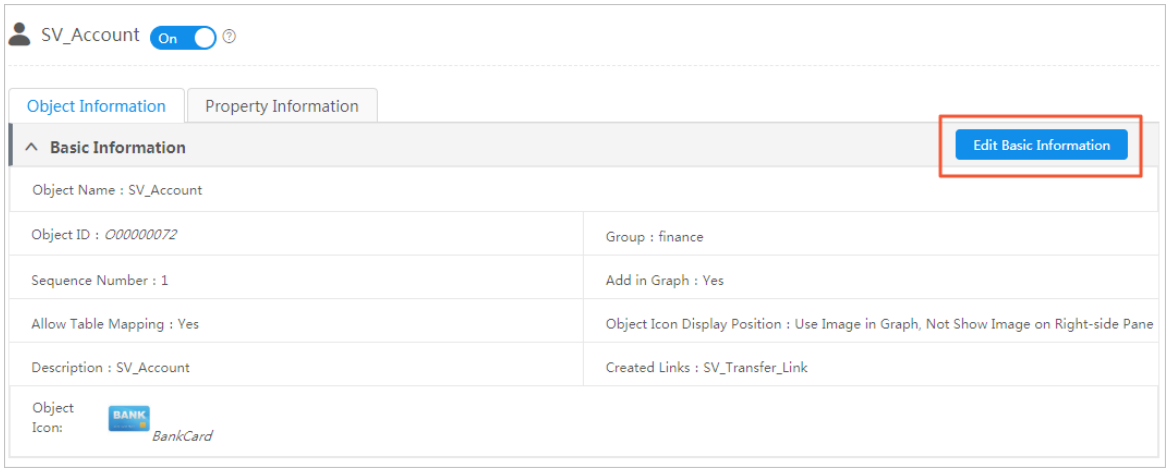
In Graph Analytics, you can modify the basic information of an object at any time, including the object name, object group, and object icon.

Prerequisites

You have created an object. For more information about how to create an object, see Create an object.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane, click the object group to which the object belongs, and then click the object to be modified.
4. In right-side area, click **Edit Basic Information**.



5. Modify the object information as needed.

   All parameters can be modified except the **Object ID**.
6. After you have configured the preceding parameters, click **Save**.

8.5.2.5. Delete an object

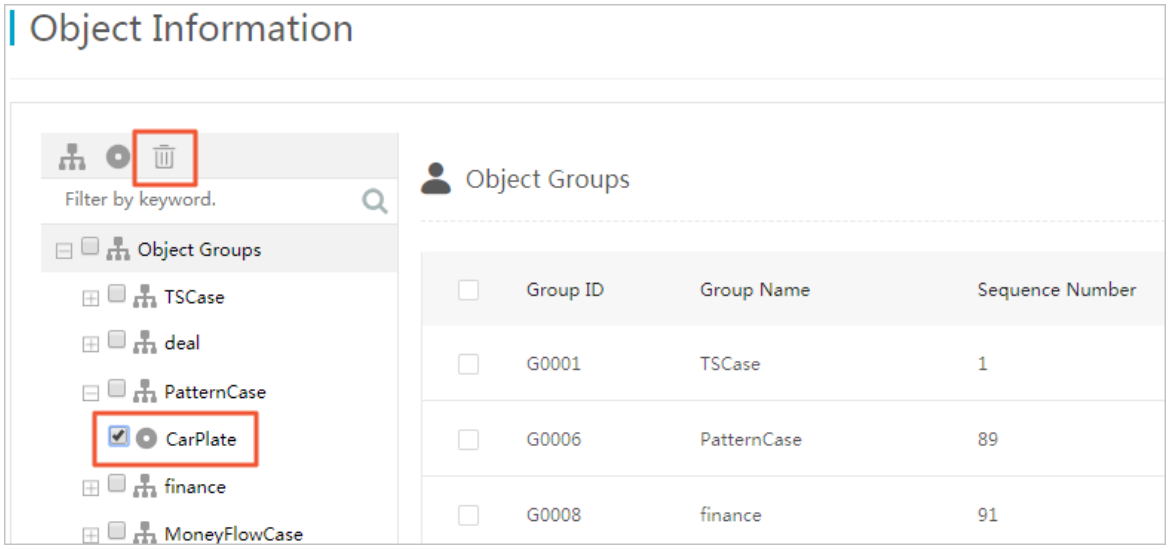
You can delete the objects that are no longer used.

Prerequisites

You have deleted the mappings between the object and the data table. You have deleted links and events that are referenced by the object.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane, click the object group to which the object belongs, and select one or more objects to be deleted. You can open multiple object groups and select objects from different groups.



4. Select the objects to be deleted, and click the icon.
5. In the **Delete Object Information** dialog box that appears, click **OK**.

8.6. Link information

8.6.1. Link groups and links

8.6.1.1. Create a link group

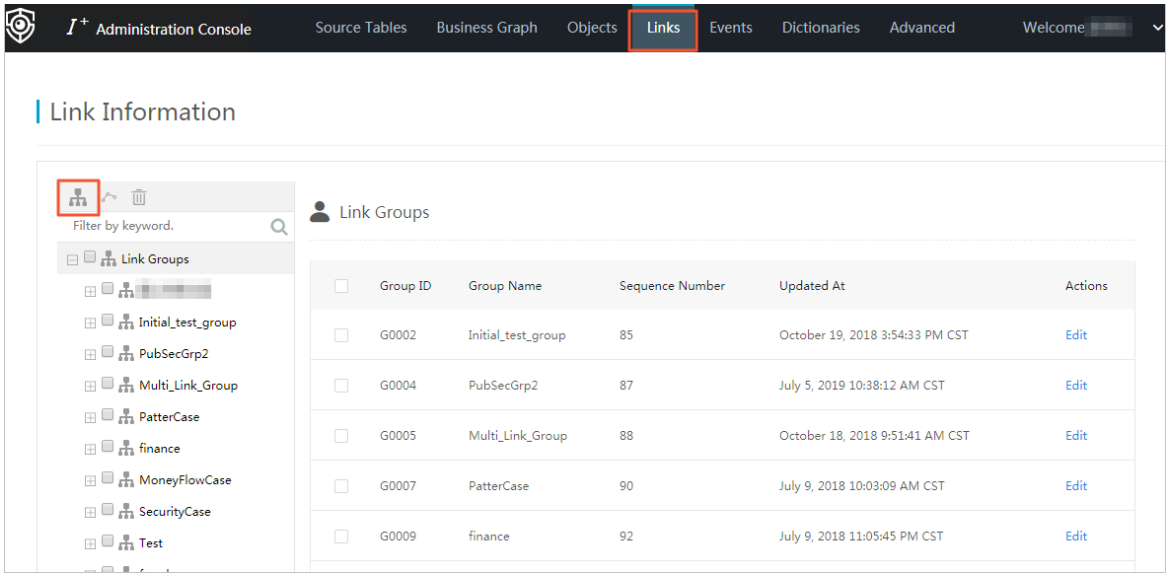
You can use link groups to classify links, so that you can search for and manage links with ease. One link must be and can be included in only one link group. You need to create a proper link group before you create a link.

Prerequisites

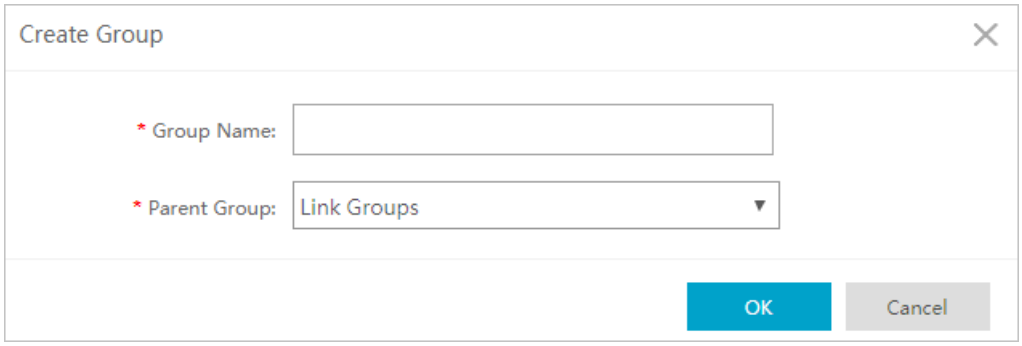
You have an account and password for Graph Analytics and have permissions to perform operations in Administration Console of Graph Analytics.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.



3. On the **Link Information** page, click the **Create Group** icon .
4. In the **Create Group** dialog box that appears, specify **Group Name** and **Parent Group**.



5. Click **OK**.

8.6.1.2. View links and link groups

Graph Analytics allows you to view all link groups in the current environment so that you can understand the existing link groups and the links under each group at any time.

Prerequisites

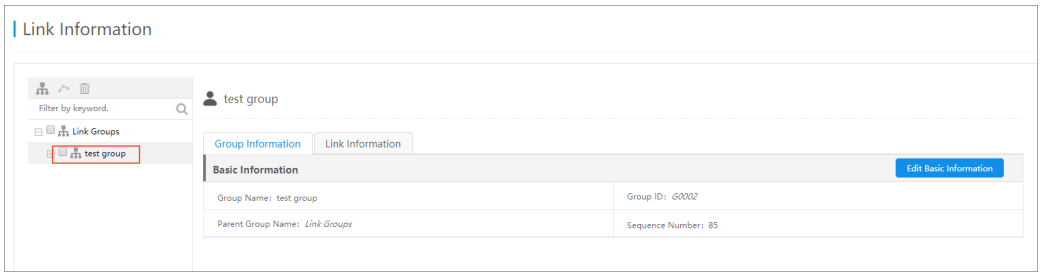
You have an account and password for Graph Analytics and have permissions to perform operations in Administration Console of Graph Analytics.

Procedure

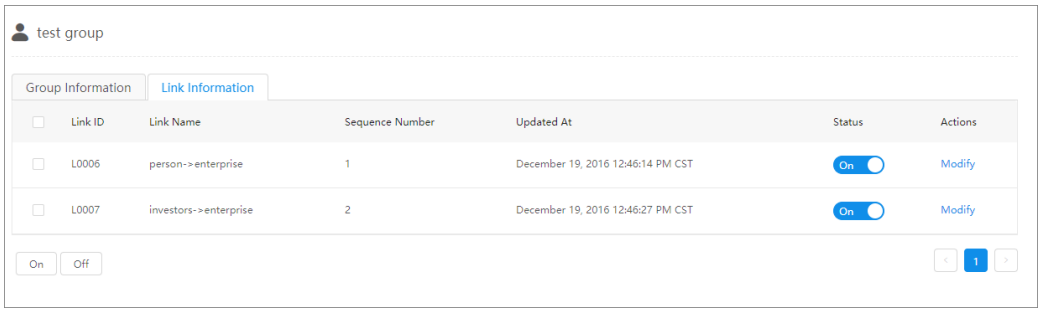
1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. In the left-side navigation pane of the **Link Information** page, click a link group. The **Group Information** and **Link Information** tabs of the link group are displayed on the right side of the page.

   The left-side navigation pane displays all link groups in the current environment. You can view the groups one by one.

Group Information



Link Information



8.6.1.3. Modify a link or link group

In Graph Analytics, you can modify the name and order number of a link or link group. You can adjust the basic information of a link or link group at any time as needed. In Graph Analytics, you can enable or disable a link.

Prerequisites

- You have created a link group. For more information about how to create a link group, see Create a link group.
- You have created a link that belongs to this link group. For more information about how to create a link, see Create a first-degree link, Create a second-degree link, or Create a multi-degree link.
- To disable links, you must first delete the dependency information of these links, including the mappings between the links and data tables.

Context

This topic describes the following operations:

- Modify the basic information about a link group
- Modify the basic information about a link
- Enable or disable a link

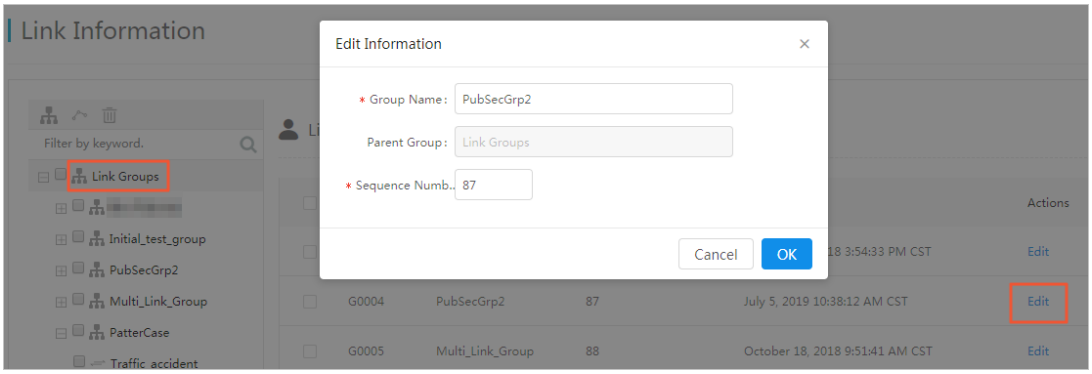
After you create a complete link by configuring the link properties), the link is automatically enabled. You can disable a link if it is no longer used for a period of time and enable it again when necessary. You cannot use a link in Analytics Workbench after the link is disabled.

Modify the basic information about a link group

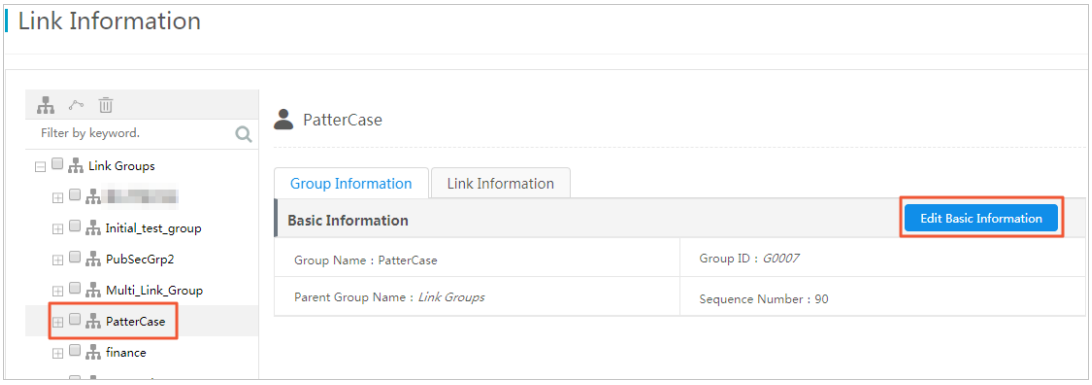
You can modify the basic information of an object group by using one of the following methods:

Configuration method	Procedure
Method 1	1. Log on to Administration Console of Graph Analytics. 2. In the top navigation bar, click **Links**. 3. In the **Link Groups** area that appears, select a link and click **Edit** in the Actions column. 4. In the **Edit Information** dialog box that appears, specify **Group Name** and **Sequence Number**, as shown in Edit the basic information. 5. Click **OK**.
Method 2	1. Log on to Administration Console of Graph Analytics. 2. In the top navigation bar, click **Links**. 3. In the left-side navigation pane, click the link group you want to modify. 4. On the right-side **Group Information** tab, click **Edit Basic Information**. Specify **Group Name** and **Sequence Number**, as shown in Edit the basic information. 5. Click **Save**.

Edit information



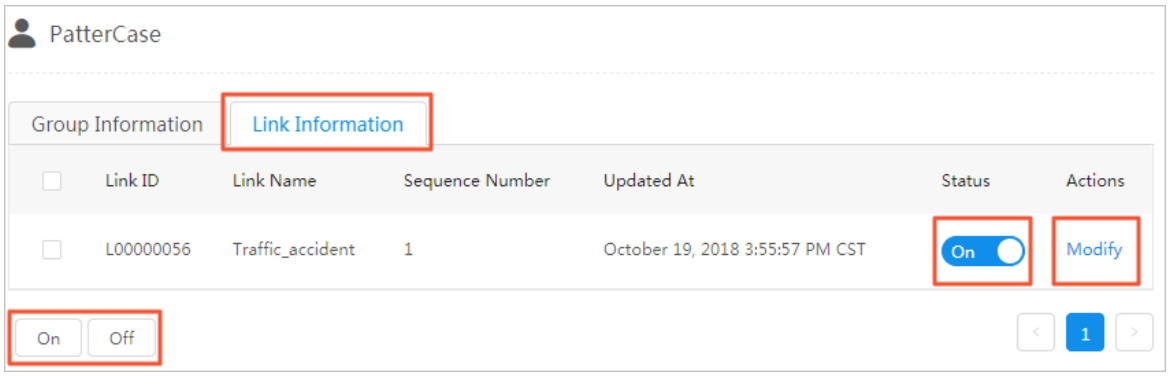
Edit the basic information



Modify the basic information about a link

1. Log on to Administration Console of Graph Analytics.

2. In the top navigation bar, click **Links**.
3. In the left-side navigation pane, click the link group to which the link belongs, and then click the **Link Information** tab.



4. In the **Edit Information** dialog box that appears, specify **Link Name** and **Sequence Number**.
5. Click **OK**.

Enable or disable a link

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. In the left-side navigation pane, click the link group to which the link belongs, and then click the **Link Information** tab on the right side of the page.
4. You can use the following methods to enable or disable a link on the **Link Information** tab.

Configuration method	Procedure
Disable a link	Disable a single link: Click the **On** icon next to the link you want to disable. Disable multiple links: Select the links you want to disable and then click **Off** in the lower part of the page. After the link is disabled, the **Status** changes from a highlighted **On** icon to a dimmed **Off** icon.
Enable a link	Enable a single link: Click the **Off** icon next to the link you want to enable. Enable multiple links: Select the links you want to enable and then click **On**. After the link is enabled, the **Status** changes from a dimmed **Off** icon to a highlighted **On** icon.

8.6.1.4. Delete a link or link group

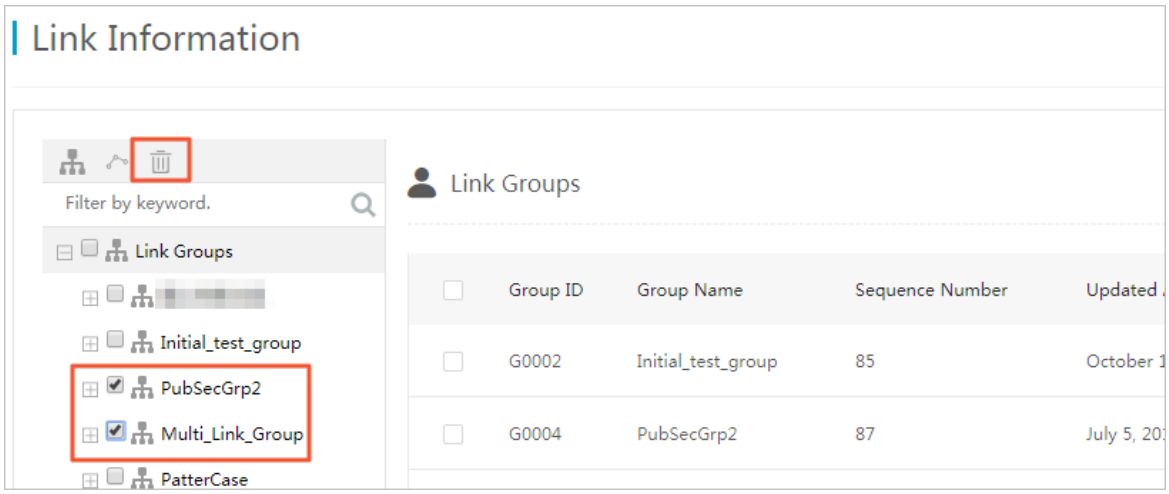
You can delete the links and link groups when they are no longer used.

Prerequisites

- Before you delete a link, you have deleted the dependency information of the link, for example, the mapping between the link and the physical table.
- Before you delete a link group, you have deleted all the links in the group. You can delete only empty link groups.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. (Optional)If there are links in the object group, you must delete these links first.
   i. In the left-side navigation pane, select all links in the link group you want to delete, and click the icon in the upper-left corner.
   ii. In the **Delete Link Information** dialog box that appears, click **OK**.
4. In the left-side navigation pane, select the link groups you want to delete, and then click the icon in the upper-left corner.



5. In the **Delete Link Information** dialog box that appears, click **OK**.

8.6.2. First-degree links

8.6.2.1. Create a first-degree link

A first-degree link reflects the direct relationship between two objects. It is the basis of second-degree links and multiple-degree links. You must create a first-degree link between objects before you perform a link analysis in Graph Analytics. A complete link contains the basic information, property information, and business parameters. This topic describes how to configure the basic information of a link.

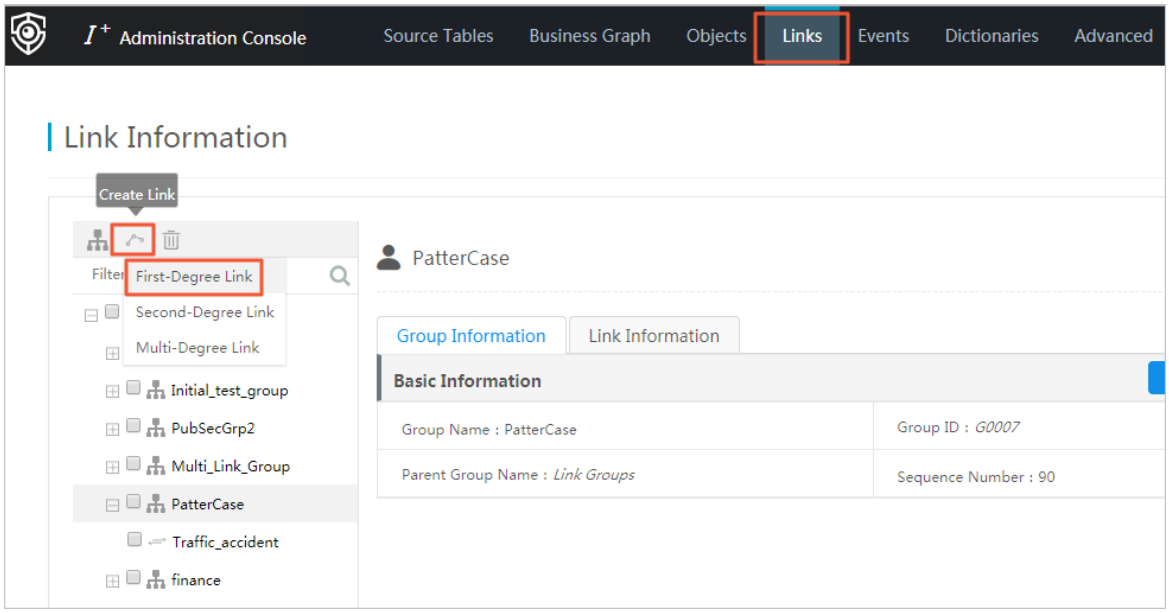
Prerequisites

- You have created and enabled a source object and a target object for the first-degree link. For more information about how to create an object, see Create an object.
- You have created a link group to which the first-degree link belongs. For more information

about how to create a link group, see Create a link group.

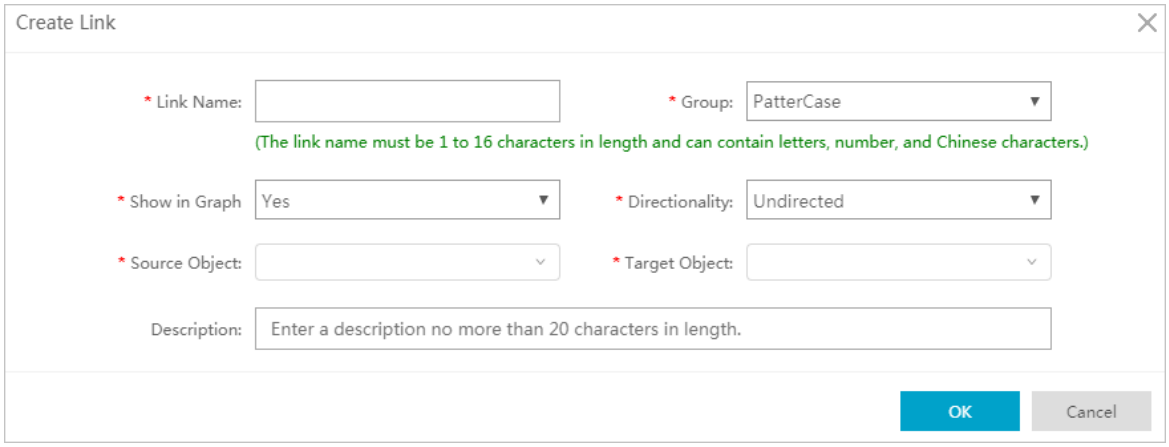
Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. The **Link Information** page appears. Click the **Create Link** icon, and click **First-Degree Link**.



4. In the **Create Link** dialog box that appears, specify the parameters.

   Parameters used to create a first-degree link describes the parameters.



Parameters used to create a first-degree link

Parameter	Description
Link Name	The user-defined link name. It must be unique.
Group	By default, the selected link group in the left-side navigation pane is used. You can also select another link group as needed.

Parameter	Description
Show in Graph	Specifies whether a link is displayed on the **Graph** page. If the value is set to **No**, this link is not displayed on the **Graph** page.
Directionality	Specifies whether the link is directional. For example, if the source object A calls the target object B, a link is established. If this link is set to be directional, it will be displayed on the **Graph** page. The link direction is from A to B (A > B).
Source Object	The source object of the first-degree link. You can select it from the drop-down list.
Target Object	The target object of the first-degree link. You can select it from the drop-down list.
Description	Enter the description of the first-degree link. This makes it easy for you to understand this link.

5. Click **OK**.

What's next

1. After you have created a first-degree link, you must specify the properties and configure the business parameters based on your requirements. For more information, see Configure link properties and business parameters.
2. To use the new link, you must log on to Analytics Workbench again.

8.6.2.2. Configure link properties and business parameters

After you add a first-degree link, you need to configure the properties and business parameters of the link based on your business requirements, so that you can view and apply this link in Analytics Workbench. This topic describes how to configure the properties and business parameters of a first-degree link.

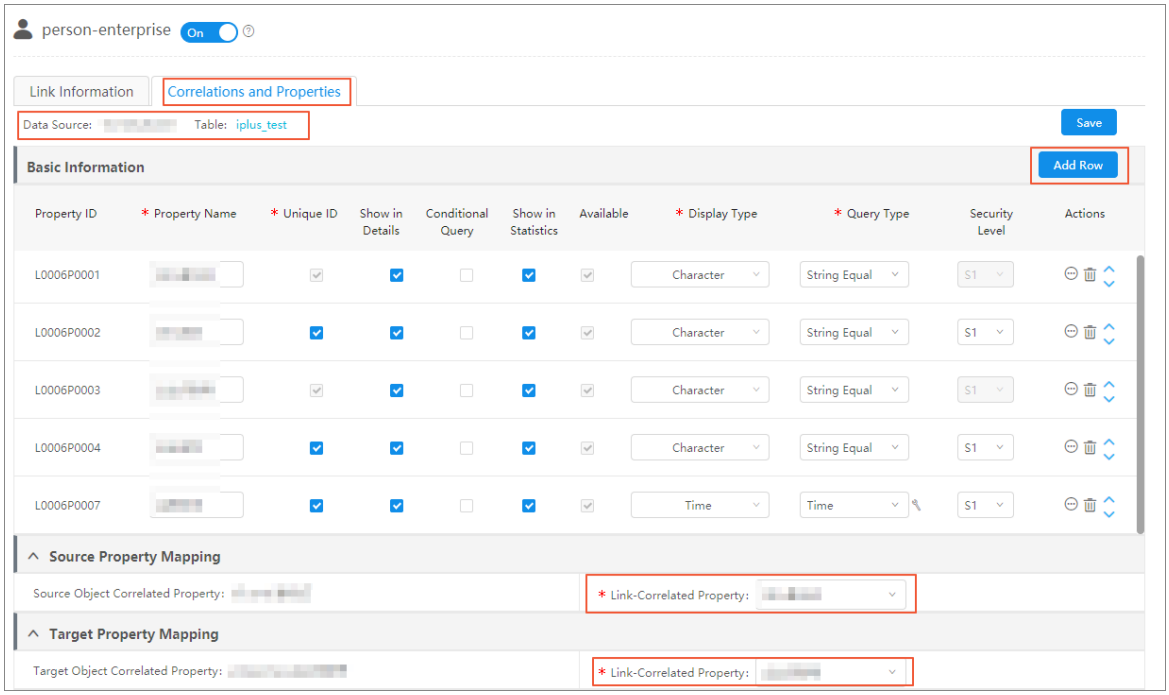
Prerequisites

You have created a first-degree link. For more information about how to create a link, see Create a first-degree link.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click **Links** on the top of the page.
3. In the left-side navigation pane, click the link group that contains the first-degree link to be configured, and then click the link.
4. In the right-side area, click the **Correlations and Properties** tab.

On the **Correlations and Properties** page, if the link has been mapped to a data table in the data source, the property information section will display **Data Source** and **Table**. You can click the table name to go to the table page.



5. Set the basic configurations in the **Basic Information** page.

   Click **Add Row** to add a property for a link in **Basic information**.

> **Note**
>
> **Link-Correlated Property** in **Source Property Mapping** and **Link-Correlated Property** in **Target Property Mapping** are not the same and they are both required to be configured. Therefore, a link must have at least two properties.

The configuration items in **Basic Information** are described in Description of basic configuration items.

Description of basic configuration items

Configuration item	Description
Property ID	The ID of a property. It is automatically generated.
Property Name	The name of the property. If you select Available, the name of the property will be displayed in Analytics Workbench.
Unique ID	Defines the logical primary key of a link table.
Show in Details	If you select this item, the property will be displayed in the **Details** tab in Analytics Workbench.

Configuration item	Description
Conditional Query	If you select this item, when you perform an analysis on the **Graph** page in Analytics Workbench, you can query a link based on the link type in **Link Type**.
Show in Statistics	If you select this item, the property is displayed in **Analytics Workbench > Graph > right-side navigation pane > Statistics**. Otherwise, it is not displayed.
Available	If you select this item, the property takes effect and is displayed in Analytics Workbench. The **Available** parameter is automatically selected for a property if any of the following parameters has been selected for the property: **Unique ID**, **Show in Details**, **Conditional Query**, and **Show in Statistics**. The **Available** parameter is automatically deselected for a property if all of the preceding parameters are deselected for the property.
Display Type	After you set the display type, the property is displayed in **Analytics Workbench > Graph > right-side navigation pane > the Details tab and the Property tab** based on the selected type. Note To display a property in the format of **Dictionary**, you need to configure a dictionary first.
Query Type	The data type that is supported in the query condition of a property. If you select **Dictionary** for **Display Type**, you must select **Dictionary Option** for **Query Type**.
Security Level	The security level for a property. A user with a lower security level cannot view the property.
Search Item Configuration	Associates this property with a search item so that the link can be searched by this property in Analytics Workbench.
Default Query Condition Settings	Specifies the default condition used for a link query. If other properties are used as conditions for the query, this condition is also included by default.
Authorization Code	After the authorization code function has been enabled, only users with the required authorization code can access this property.
Derived Property	Sets a property as a derived property so that it can be generated automatically based on other properties. You can set the derivative method based on your requirements.
Move Up and Move Down arrows	The Move Up arrow and the Move Down arrow can be used to adjust the order of properties that are displayed in Analytics Workbench.

6. You can set **Link-Correlated Property** in **Source Property Mapping** and **Target Property Mapping**.

These two configurations are related with the Source Object and the Target Object of the link.

Take **Source Property Mapping** as an example: **Source Object Correlated Property** and **Link-Correlated Property** must be mapped to the same column in the same table. The **Source Object Correlated Property** parameter is the primary key property of the source object, which is automatically loaded according to the Source Object parameter. For the **Link-Correlated Property** parameter, you must select the link property that is mapped to the same column in the same table as the Source Object primary key.

Set **Target Property Mapping** in the same way you set **Source Property Mapping**.

7. (Optional)Set **Advanced**, **Accumulative Statistics Settings**, and **Link Weight Settings** based on your requirements.

   For more information about the configurations of key parameters, see Parameter configurations.

Parameter configurations

Category	Configuration item	Description
Advanced	**Chronological Time Property**	Specifies the link properties based on which chronological analysis is performed. From the drop-down list, select one or more link properties of which the query type is time.
	Time Property for Behavior Analysis	Specifies the link properties based on which behavior analysis is performed. From the drop-down list, select one or more link properties of which the query type is time.
	Linked Times	Specifies the property of which the number of the same values are counted. The total number is displayed as the number of link occurrences. The **Linked Times** parameter is used as the default setting to filter link types. For example, if there are two limes of A > C calls in the call log, the analysis result displays that the number of A > C calls is two.
	Details Sorting Property	Specifies the property by which the returned behavior details are sorted by default.
Accumulative Statistics Settings	N/A	Used to perform logical statistics for link properties of which the query type is numeric range. The logical statistics operations include top, =, and >=. This configuration applies to business scenarios where statistics filtering is required for link query results. The **Linked Times** parameter is used to filter records in link query results. You can add statistical conditions to filter the link properties of which the query type is numeric range.

Category	Configuration item	Description
Link Weight Settings	N/A	You can specify a link property of which the query type is numeric range and calculate the link weight based on the numeric range specified for the link property.

8. After you have modified the parameters, click Save. A message is displayed, indicating that the modifications have been saved.

8.6.3. Create a second-degree link

A second-degree link reflects the relationship between two objects established by using an intermediary object. Unlike a first-degree link, a second-degree link can be used to explore a more complex relationship network among objects.

Prerequisites

- Related first-degree links have been created for the second-degree link. For more information about how to create a first-degree link, see Create a first-degree link.
- A link group has been created for the second-degree link. For more information about how to create a link group, see Create a link group.

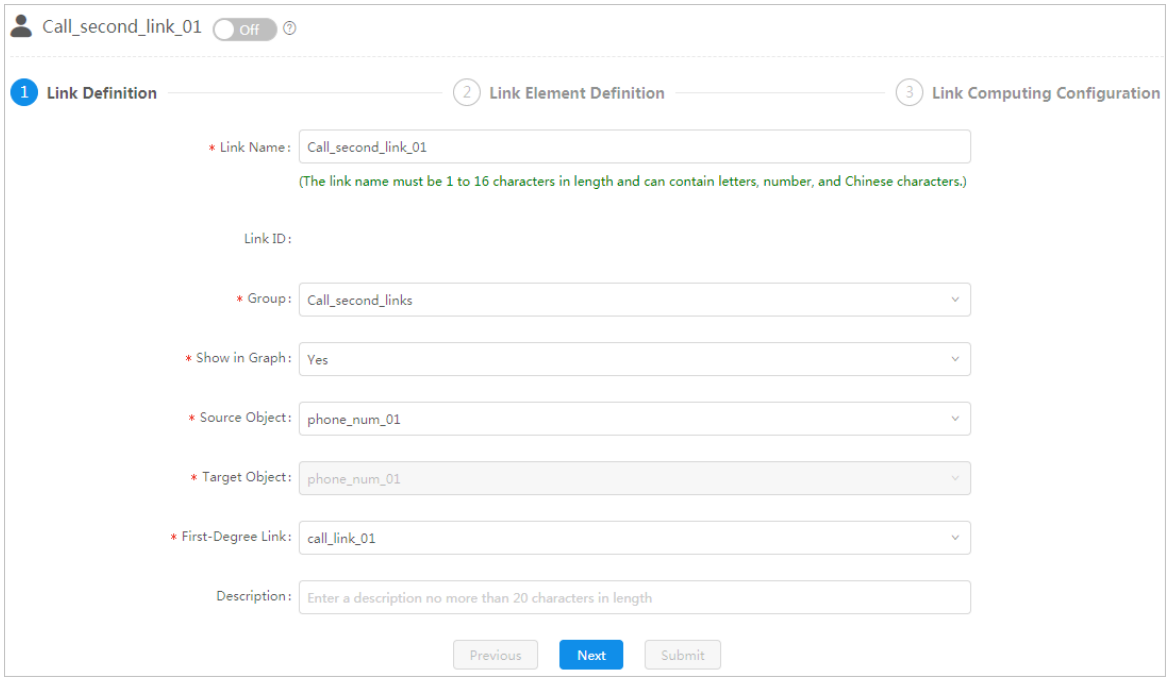
Context

A second-degree link is created based on first-degree links and can be split into two first-degree links. For example, object A has a first-degree link with object C, and object B also has a first-degree link with object C. In this case, you can establish a second-degree link between object A and object B. Typically, people taking the same train or plane or staying in the same hotel have a second-degree link and can be analyzed by using the second-degree link model.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. On the **Link Information** page that appears, click the **Create Link** icon and select **Second-Degree Link**.
4. Configure the parameters on the **Link Definition** page.

   Parameters describes the parameters.

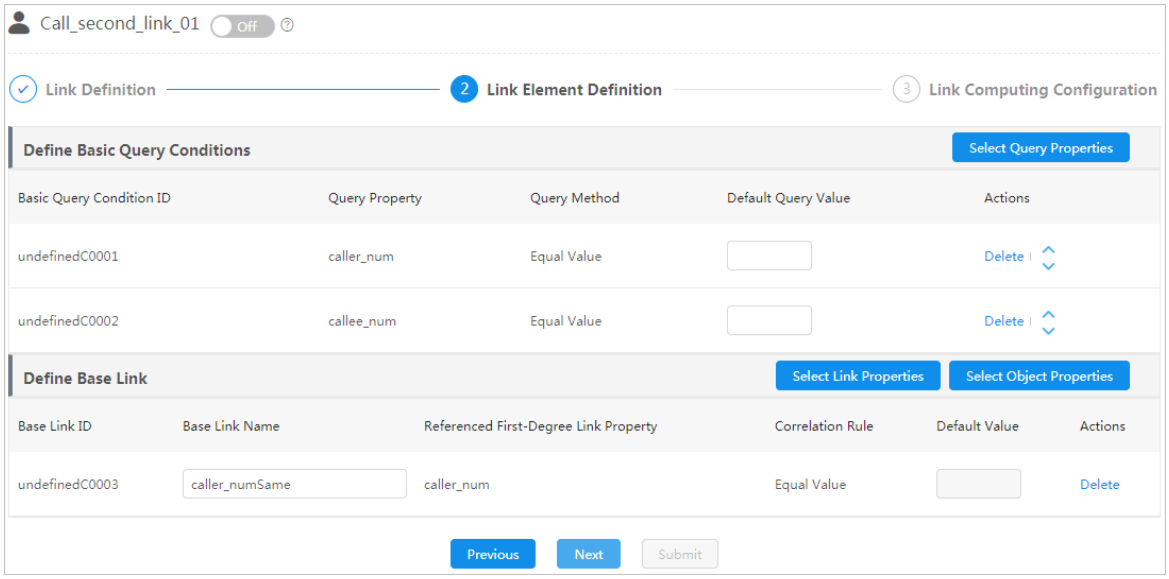


Parameters

Parameter	Description
Link Name	The user-defined link name. It must be unique.
Group	By default, the selected link group in the left-side navigation pane is used. You can also select another link group as needed.
Show in Graph	Indicates whether a link can be displayed on the **Graph** page. If this parameter is set to **No**, this link is not displayed on the **Graph** page.
Source Object	The source object of the second-degree link.
Target Object	The target object of the second-degree link.
First-Degree Link	The first-degree links between the source object and the target object. These links can be used to create a second-degree link.
Description	The description of the second-degree link that helps you understand the link.

5. After you have configured the preceding parameters, click **Next** to configure the parameters in **Link Element Definition**.

   Parameters describes the parameters.

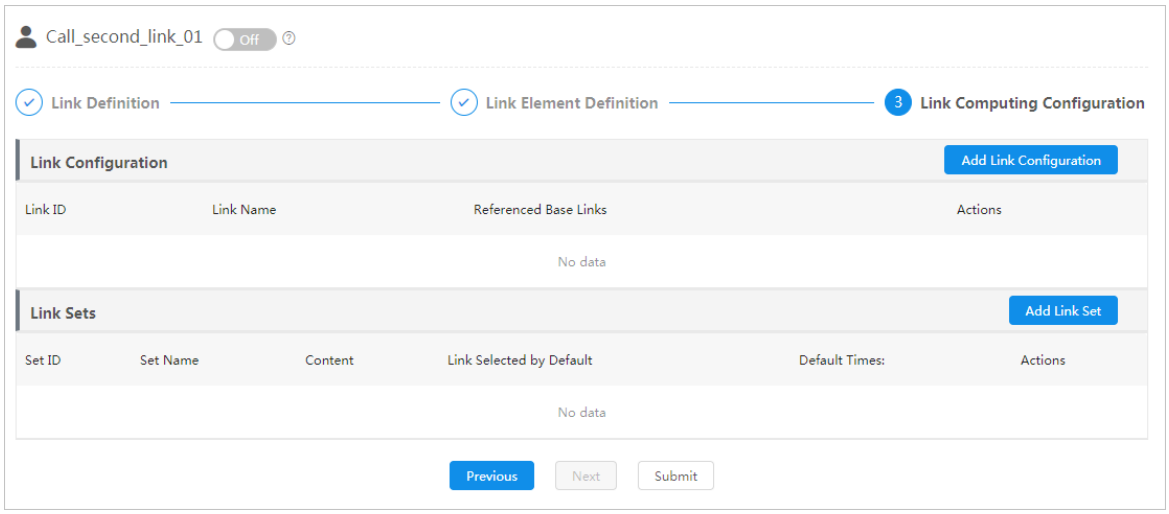


Parameters

Section	Parameter	Description
Define Basic Query Conditions	Select Query Properties	Allows you to define the basic query conditions that can be used to query a link. You can view the conditions in the **Basic Properties** section in the **Link Type** settings when you configure link extension.
Define Basic Link	Select Link Properties and Select Object Properties	Allows you to define the table columns that are needed to perform a query. You can define the alias of a column in **Base Link Name**. The base link can be referenced in **Link Configuration** in the next step. Configure the Correlation Rule. The Default Value will be used when you perform a query in Analytics Workbench.

6. After you have configured these parameters, click **Next** and then configure the parameters in **Link Computing Configuration**.

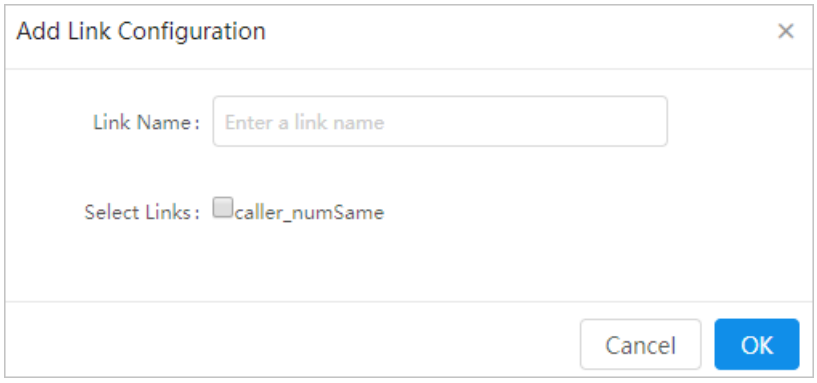
   Parameters describes the parameters.



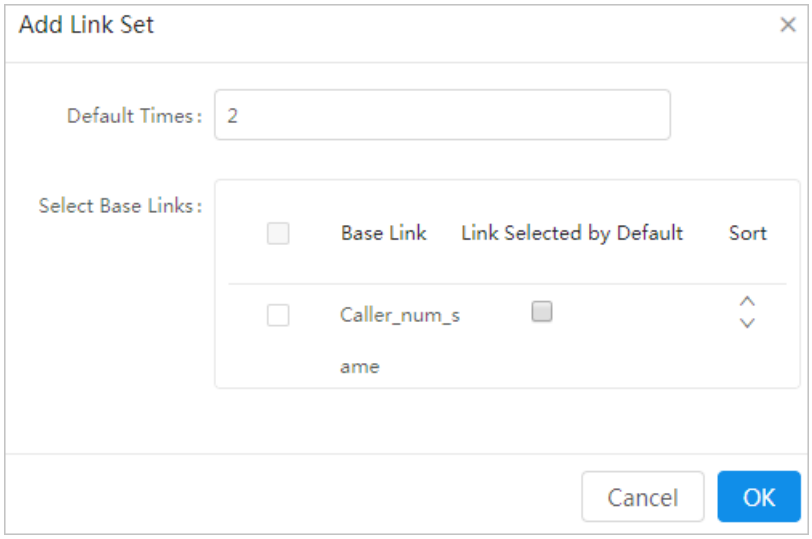
Parameters

Section	Parameter	Description
Link Configuration	**Add Link Configuration**	Allows you to combine multiple first-degree links to form a link that meets multiple conditions. i. Click **Add Link Configuration**, specify **Link Name**, and then select links. ii. Click **OK** to add a link that meets multiple conditions. iii. You can repeat the preceding steps to create more links of the same type as needed.
Link Sets	**Add Link Set**	A link set is a group of links configured in **Link Configuration**. Such link sets define the advanced query conditions for link queries in link analysis. i. Click **Add Link Set** and configure the parameters. Key parameters to add a link set: ■ **Default Times**: the minimum number of occurrences of a link set that can be counted as a query match. The default value is 2. ■ **Base Link**: the links displayed in the Select Base Links area are all the links that have been configured in **Link Configuration**. Each link can be contained in only one link set. ■ **Link Selected by Default**: the default query condition that is displayed on Analytics Workbench. If a link set that contains multiple links is used for a link query, each analysis is performed based on one of the links. By default, **Link Selected by Default** is used. ii. Click **OK**. iii. You can repeat the preceding steps to create more link sets as needed.

Add a link configuration



Add a link set



7. After you have configured the preceding parameters, click **Submit** to create a second-degree link.

8.6.4. Create a multi-degree link

A multi-degree link reflects the relationship between two objects established by using multiple intermediary objects. Compared to a first-degree link or a second-degree link, a multi-degree link can further explore the complex relationship between objects.

Prerequisites

- You have created first-degree links for the multi-degree link. For more information about how to create a first-degree link, see Create a first-degree link.
- You have created second-degree links for the multi-degree link. For more information about how to create a second-degree link, see Create a second-degree link.
- You have created a link group for the multi-degree link. For more information about how to create a link group, see Create a link group.

Context

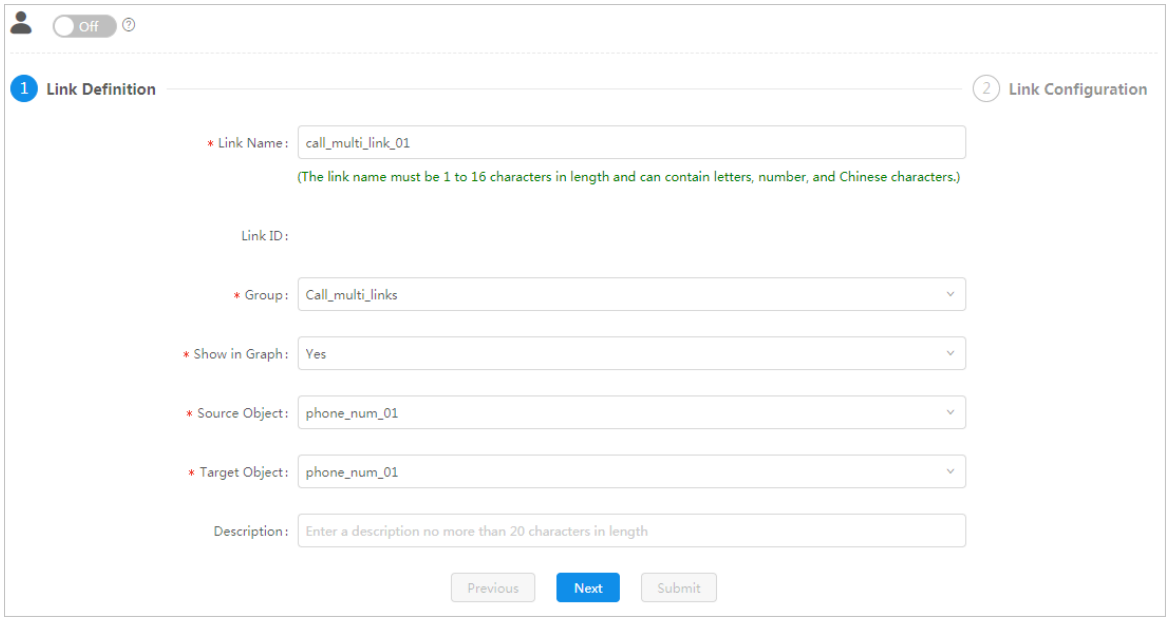
A multi-degree link uses multiple first-degree links and second-degree links to query the relationship between two objects.

For example, if mobile phone A and mobile phone B do not have call or text message records and only have email correspondence records, there are only indirect links between the two mobile phones. In this example, there is no direct relationship between mobile phone A and mobile phone B, so they cannot be directly linked. When mobile phone A sends an email to mobile phone B, an indirect link is established. This indirect link involves three first-level links: the link between mobile phone A and mailbox A, the link between mailbox A and mailbox B, and the link between mobile phone B and mailbox B. You can use mobile phone A to query mobile phone B by using the three links. Indirect links can also be established between two mobile phones that are communicated by using apps such as Facebook and Twitter.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Links**.
3. On the **Link Information** page that appears, click the **Create Link** icon and select **Multi-Degree Link**.
4. Configure the parameters on the **Link Definition** page.

   Link definition parameters describes the parameters.

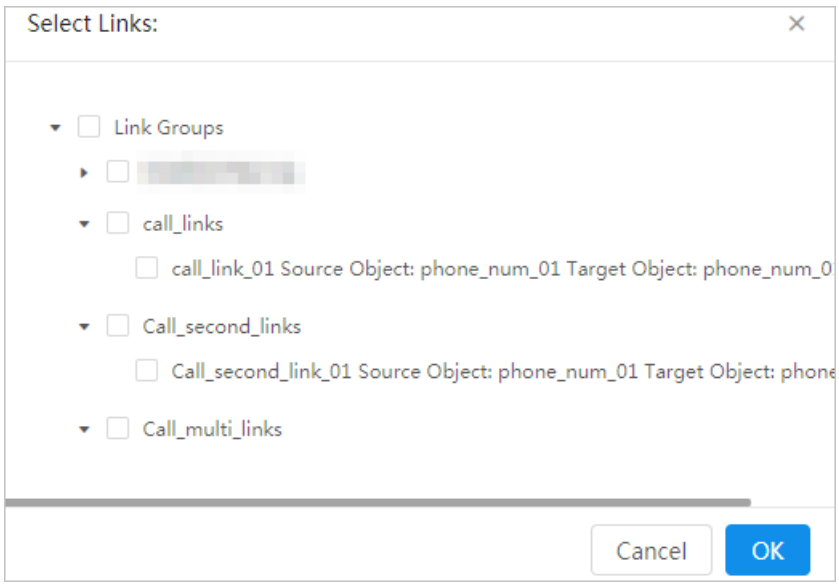


Link definition parameters

Parameter	Description
Link Name	The user-defined link name. It must be unique.
Group	By default, the selected link group in the left-side navigation pane is used. You can also select another link group as needed.
Show in Graph	Specifies whether a link is displayed on the **Graph** page. If the value is set to **No**, this link definition is not displayed on the **Graph** page.
Source Object	The source object of the multi-degree link. You can select it from the drop-down list.

Parameter	Description
Target Object	The target object of the multi-degree link. You can select it from the drop-down list.
Description	The description of the multi-degree link that helps you understand the link.

5. Click **Next**.
6. On the **Link Configuration** page, click **Select Links** in the upper-right corner, and then select the related first-degree links and second-degree links.

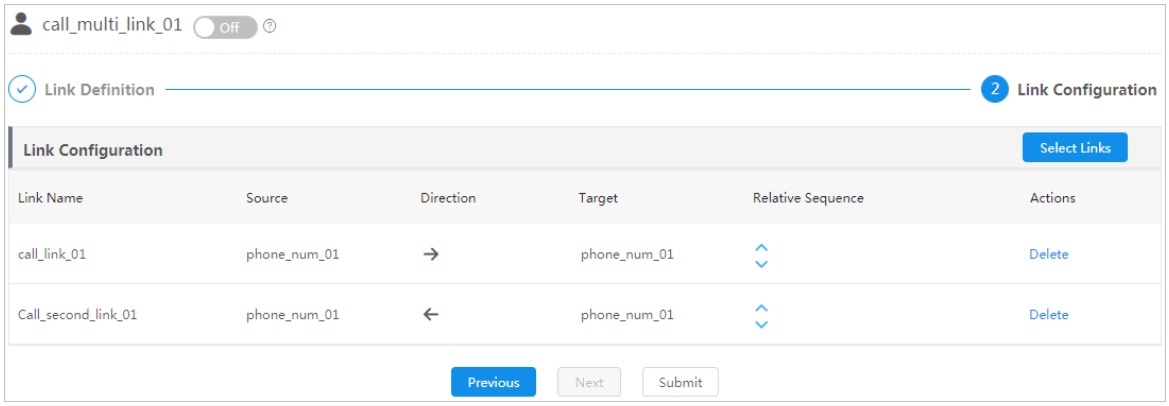


7. After you have configured the preceding parameters, click **OK** to add first-degree links and second-degree links.

   In most cases, you may need to add two first-degree or second-degree links with opposite query directions. For example, if you want to send an email by using your mobile phone, the query direction is from the mobile phone to the email box. If you want to receive an email on your mobile phone, the query direction is from the email box to your mobile phone. Therefore, two first-degree links can be added with opposite query directions.

8. On the **Link Configuration** page, click the right or left arrow in the **Query Direction** column to adjust the query direction.

   The source object and the target object must be correlated with each other by using intermediary objects..



9. After you have configured the preceding parameters, click **Submit** to add a multi-degree link.

8.7. Event information

8.7.1. Event groups

8.7.1.1. Create an event group

You can use event groups to classify events, so that you can easily find and manage events. Any event must be and can only be grouped into one event group. You need to create a proper event group before you create an event.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. At the top of the left-side navigation pane, click the **Create Group** icon . The **Create Group** dialog box appears.
4. Enter a **Group Name** as needed, and then select **Event Groups** from the *Parent Group* drop-down list.
5. Click **OK**.

8.7.1.2. View an event group

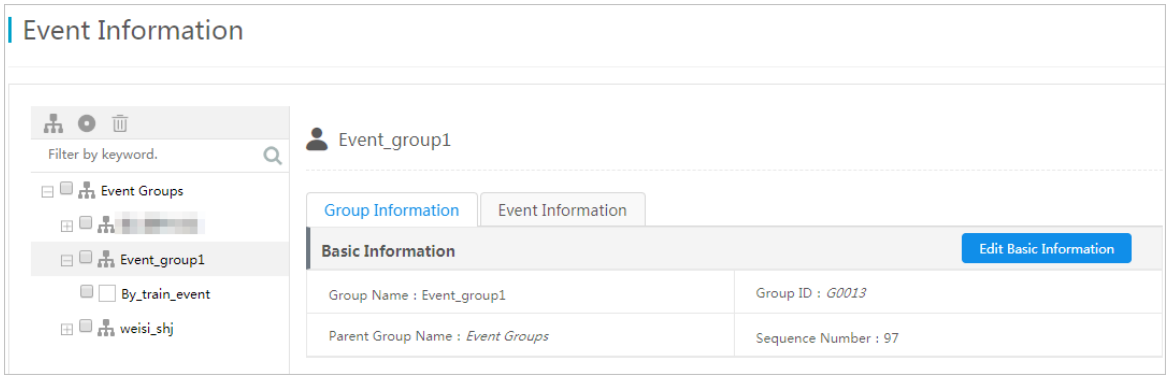
You can view the basic information of an event group and the events in the group in Administration Console.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. From the left-side navigation pane, select an event group such as **By Train**. The detailed information about this event group is displayed on the right side of the page.



You can view event group information on the **Group Information** and **Event Information** tabs. The **Group Information** tab displays the basic information about the event group. The **Event Information** tab displays all events in the group.

8.7.1.3. Modify an event group

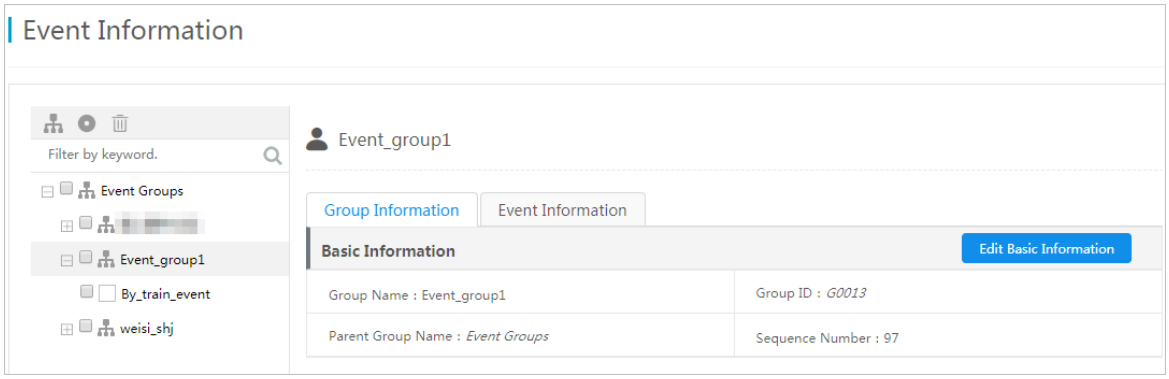
You can modify the basic information of an event group and the events in the group in Administration Console.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. From the left-side navigation pane, select an event group such as **By Train**. The detailed information about this event group is displayed on the right side of the page.



4. Click **Edit Basic Information** on the **Group Information** page. You can then modify the **Group Name** and **Sequence Number** of the group.
5. Click **Save**.

8.7.1.4. Delete an event group

If an event group is no longer used, you can first delete all events in the group and then delete the event group.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have deleted all events in the event group. For more information about how to delete an event, see Delete an event.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. Select the event groups that you want to delete from the left-side navigation pane, and then click the **Delete** icon on the top of the pane.
4. In the message box that appears, click **OK**.

8.7.2. Events

8.7.2.1. Create an event

Events are used to analyze the behaviors of objects. Event information is used to define the event models in Graph Analytics. A complete event contains basic event information, properties, and relevant parameters. This topic describes how to configure the basic information of an event.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created an event group for the event. For more information about how to create an event group, see Create an event group.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. On the top of the left-side navigation pane, click the **Create Event** icon. Set the parameters in the **Create Event** dialog box. For more information about the parameters and descriptions, see Parameters of created events.

Parameters of created events

Parameter	Configuration method

Parameter	Configuration method
Event Name	The name of the event. Set this parameter as needed. Each event name must be from 1 to 16 characters in length, and can contain letters, digits, and Chinese characters.
Event Description	The description of the event. Set this parameter as needed to help users understand the event.
Group	The event group that the event belongs to. Set this parameter as needed.
Event Icon	The event icon. You can select an icon in **Icon Library**, or enter an accessible URL to reference an external icon.

4. After you set the parameters, click **OK**. A message is displayed, indicating that the event has been created.

What's next

1. After you have created an event, you must configure the properties and business parameters based on your business requirements. For more information, see Configure event property parameters.
2. After you have configured the properties and business parameters of an event, you must log on to Analytics Workbench again to use the new event.

8.7.2.2. Configure event property parameters

After you have created an event, you must configure the event properties. Event properties are critical to an event. You can configure event properties, and correlate the properties to objects on the Property Information tab.

Prerequisites

Make sure that you have obtained an account and a password for Graph Analytics and you have been authorized with the required **Event Permissions**.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. Click an event in the left-side navigation pane, and click the **Property Information** tab on the right side of the page. The **Property Information** tab appears.
4. Set the required event parameters. The required parameters are displayed in the **Basic Information** and **Set Mappings Between Correlated Objects and Properties** areas. The event parameters are described in Required event property parameters.

> **Note** To save the property information, you must set all the required parameters.

Required event property parameters

Area	Parameter	Description
	Property ID	The ID of a property. It is automatically generated.
	Property Name	The property name that is displayed on Analytics Workbench. We recommend that you enter a name that describes the business type.
	Primary Key	Each primary key uniquely identifies an event. A property cannot be deleted after it has been configured as a primary key.
	Show in Graph	Note For each event, you must select at least one property to be displayed in the graph. If you select this parameter, the **Property Name** of this property will be displayed in **Graph** in Analytics Workbench together with the object node. Otherwise, the property is not displayed. For example, if this parameter is selected by the ID card property of an event, the ID card number will be displayed in **Graph** with the event. If the name property is also selected, the name and the ID card number will be displayed with the event. If you select this parameter for a specific property, a bubble icon appears next to the option. You can click the bubble icon to set whether to show the **Property Name** in **Graph**. For example, if you select this parameter for an ID card number, and set to display the **Property Name** in **Graph**, the ID card property displayed in **Graph** will be **ID card number: xxxxxx**.
	Show in Properties	If you select this parameter for a property, the property will be displayed in the right-side pane on the **Graph** page of Analytics Workbench. To view the property, choose **Properties** > **Event Properties**.
	Element Identifier	The element identifier of a property. Set this parameter based on the actual property.
	Conditional Query	If you select this parameter for a property, the event can be queried based on this property in **Link Type** on the **Graph** page of Analytics Workbench when you perform a relationship analysis.

Area	Parameter	Description
Basic Information	Show in Statistics	If you select this parameter for a property, the event will be displayed in the right-side pane on the **Graph** page of Analytics Workbench. To view the property, choose **Statistics** > **Event Distribution**.
	Available	If you select this parameter for a property, the property takes effect and can be displayed on the Graph page of Analytics Workbench. This parameter is selected by default and cannot be changed.
	Display Type	The format in which a property is displayed in the right-side pane of the **Graph** page on Analytics Workbench. Set this parameter as needed.
	Query Type	The data type that is supported in the query condition of a property.
	Security Level	The security level for a property. A user with a lower security level cannot view the property.
	Search Item Configuration	Click the **More** icon in the Actions column corresponding to a property. Set the following four parameters: ○ Search Item Configuration: Search items are displayed in the drop-down list only after they have been configured in Configure a search item. ○ Default Query Condition Settings: The default condition used for a link query. If other properties are used as conditions for a query, this condition is also included by default. ○ Authorization Code: After the authorization code function has been enabled, only authorized users can access this property. ○ Derived Property: After a property is set as a derived property, it can be generated automatically based on other properties. Configure the method in which the column is generated based on your needs.
	Default Query Condition Settings	
	Authorization Code	
	Derived Property	
	Delete	When a property is no longer used, you can click the **Delete** icon to delete this property.

Area	Parameter	Description
Set Mappings Between Correlated Objects and Properties	Add Correlated Object	Adds a mapping between an object and the event. One event must have at least two objects mapped to it. Click **Add Correlated Object** to add a correlated object, and configure the mapping between the event and the primary keys of the object you have added.

5. (Optional)After you have set the required parameters, you can set the optional parameters as needed.

   The optional parameters are included in the **Advanced** and **Display Settings** areas. The optional parameters are described in Optional event property parameters.

   > **Note** **Location Settings** are currently not supported.

Optional event property parameters

Area	Parameter	Description
Advanced	Behavior Property	Defines the properties based on which a behavior analysis is performed.
	Default Details Sorting Property	Defines the property by which the details are sorted.
	Logical Relation of Authorized Properties	The logical relationship between the authorization codes of properties in each record. ○ AND: The current record is visible only to the users who meet all authorization code conditions of the properties in this record. ○ OR: The current record is visible to the users who meet any one authorization code condition of the properties in this record.
Display Settings	Enable Display	Indicates whether to show the event details.
	Group-by Properties	Indicates the property based on which events are aggregated. For example, aggregate **Travel Events** into a folder based on the **Train Number** property.

6. After you have completed the configurations, click **Save** in the upper-right corner. A success message is displayed after the modifications have been saved.

   An event is automatically enabled after its properties have been saved.

8.7.2.3. Enable and disable an event

If you do not need to use a specific event during a specific period, you can disable this event, and this event can be re-enabled.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- To disable some events, you must first delete the dependency information of these events, including the mappings between the events and data tables and the objects referenced by the events.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. You can enable or disable an event by using one of the following methods.

Method	Operation
Operations on the Event Information page	In the left-side navigation pane, select an event to be enabled or disabled, and click the toggle switch to **enable** or **disable** the event.
Operations in Event Groups	i. In the left-side navigation pane, select the event group that contains the event you want to enable or disable. Click the **Event Information** tab to go to the **Event Information** page. ii. Click the toggle switch to **enable** or **disable** the event. You can also select the events to be enabled or disabled, and click the **Off** button or the **On** button at the bottom of the page to disable or enable multiple events at a time.

8.7.2.4. View an event

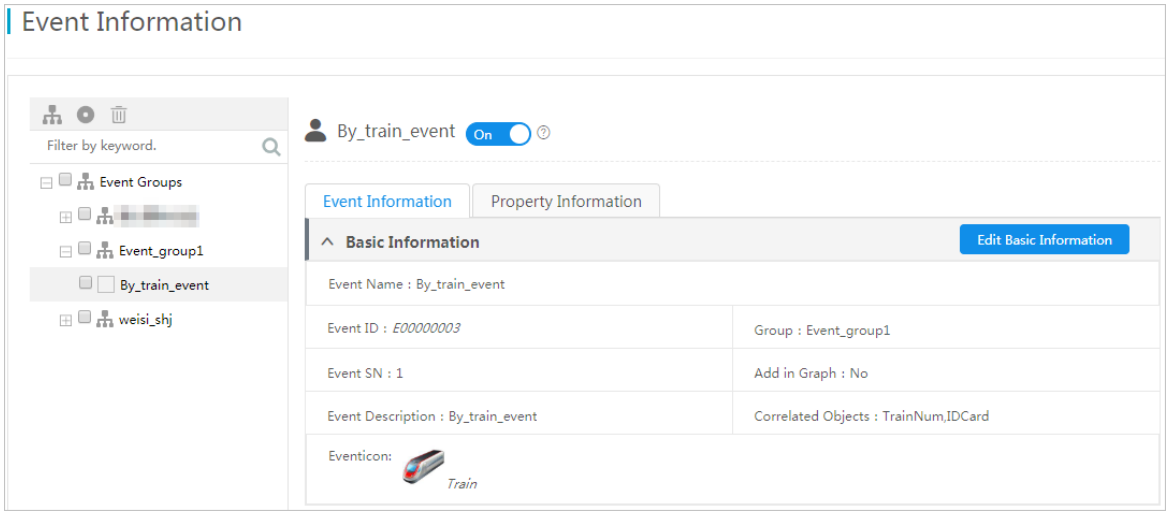
After you have configured an event, you can view the newly created event and all the events that have been created.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. In the left-side navigation pane, select an event and view the event details on the right side of the page.



Two tabs are displayed: The **Event Information** tab and the **Property Information** tab. The **Event Information** tab displays the basic information of the event. The **Property Information** tab displays the properties, correlated objects and property mappings, advanced settings, location settings, and display settings.

8.7.2.5. Modify an event

You can modify the basic information of the event you have created based on your needs.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. In the left-side navigation pane, select an event to be modified, and click **Edit Basic Information** to modify the parameters based on your requirements. The parameter configurations are described in Modify parameter configurations of events.

Modify parameter configurations of events

Parameter	Description
Event Name, Group, Event Description, and Event Icon	For more information about parameter descriptions, see Create an event.
Event SN	You can change the event sequence number based on your requirements.
Add in Graph	This parameter is not in use. You do not need to configure it.

4. After you have modified the parameters, click **Save**. A message is displayed, indicating that the modifications have been saved.

8.7.2.6. Delete an event

You can delete an event that is no longer used.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Click the **Events** tab in the top navigation bar. The **Event Information** page appears.
3. In the left-side navigation pane, select the events to be deleted, and click the **Delete** icon on the top.
4. In the dialog box that appears, click **OK**. A message is displayed, indicating that the selected events have been deleted.

8.8. View the business graph

You can directly view all configured link models on the Business Graph page. Solid lines indicate first-degree links, and dotted lines indicate second-degree or multiple-degree links.

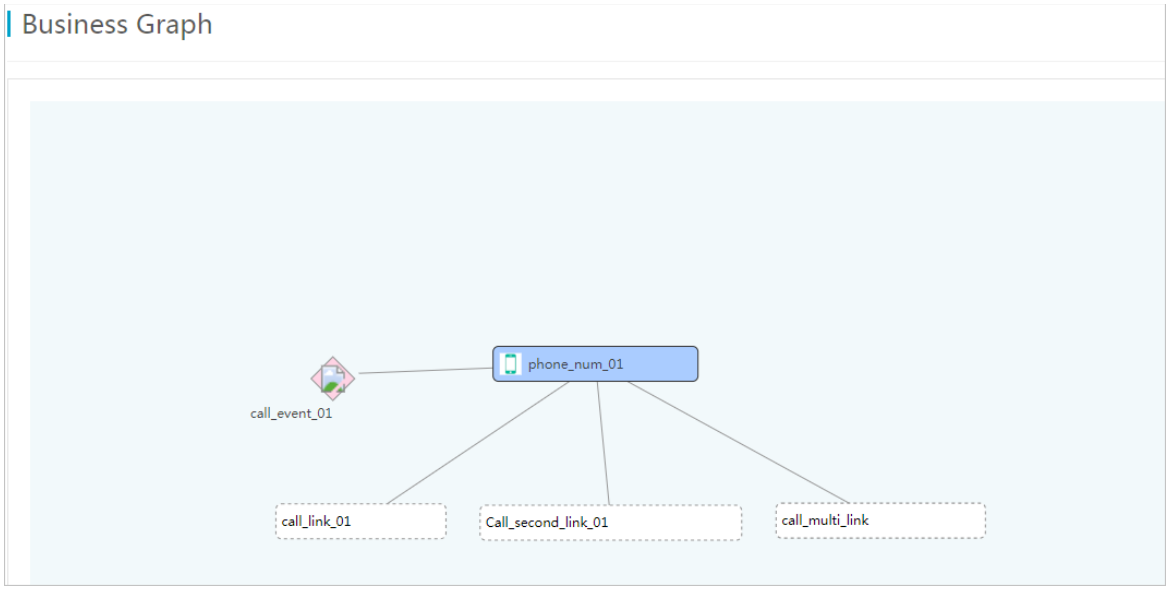
Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Business Graph**.

   On the **Business Graph** page, you can view the created link models between objects, links, and events.



8.9. Advanced configurations

8.9.1. Manage a system model

The Import Models page allows you to configure system models that are used to import data to Analytics Workbench.

Prerequisites

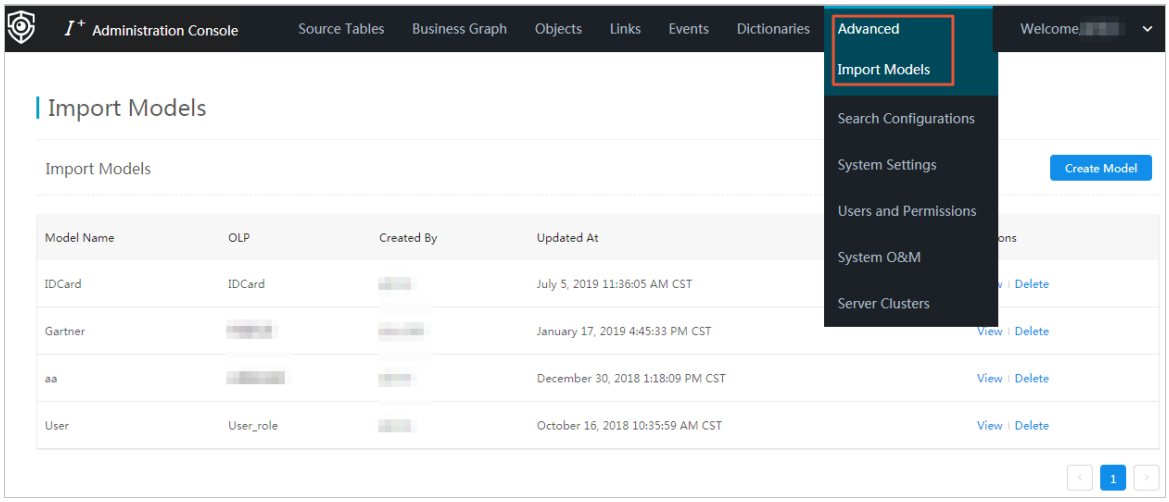
Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Context

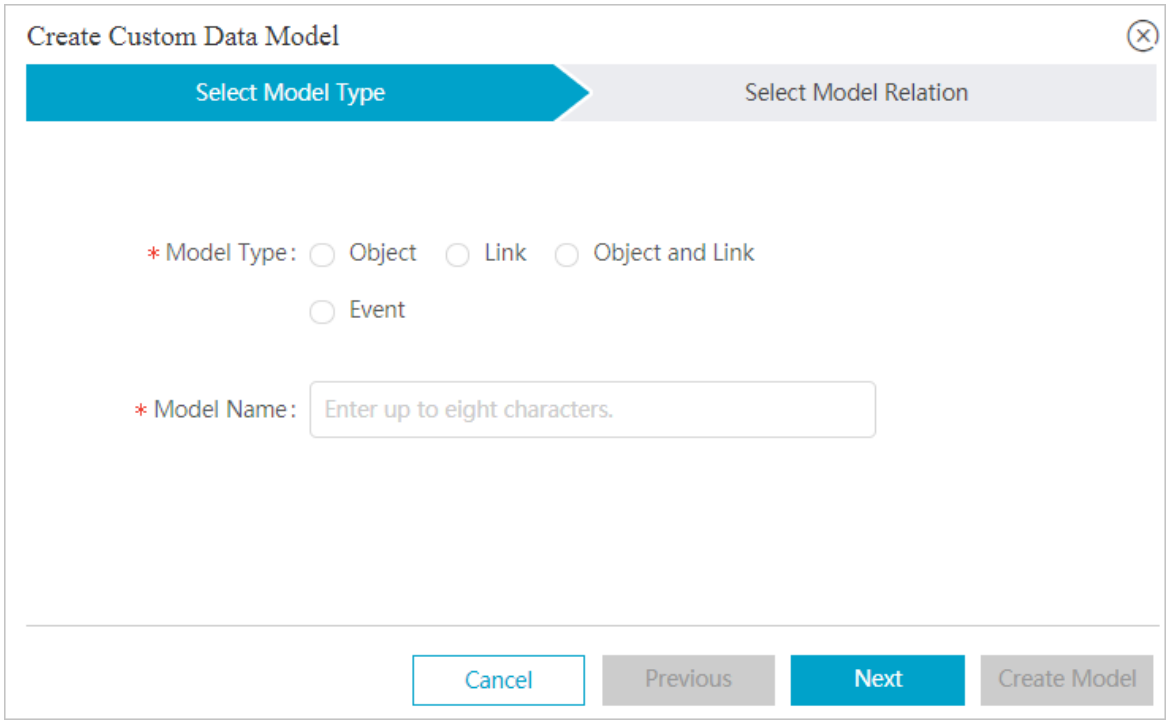
You can create, view, and delete models on the Import Models page. This topic provides examples to help you learn more about these operations.

Create a model

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Import Models**.

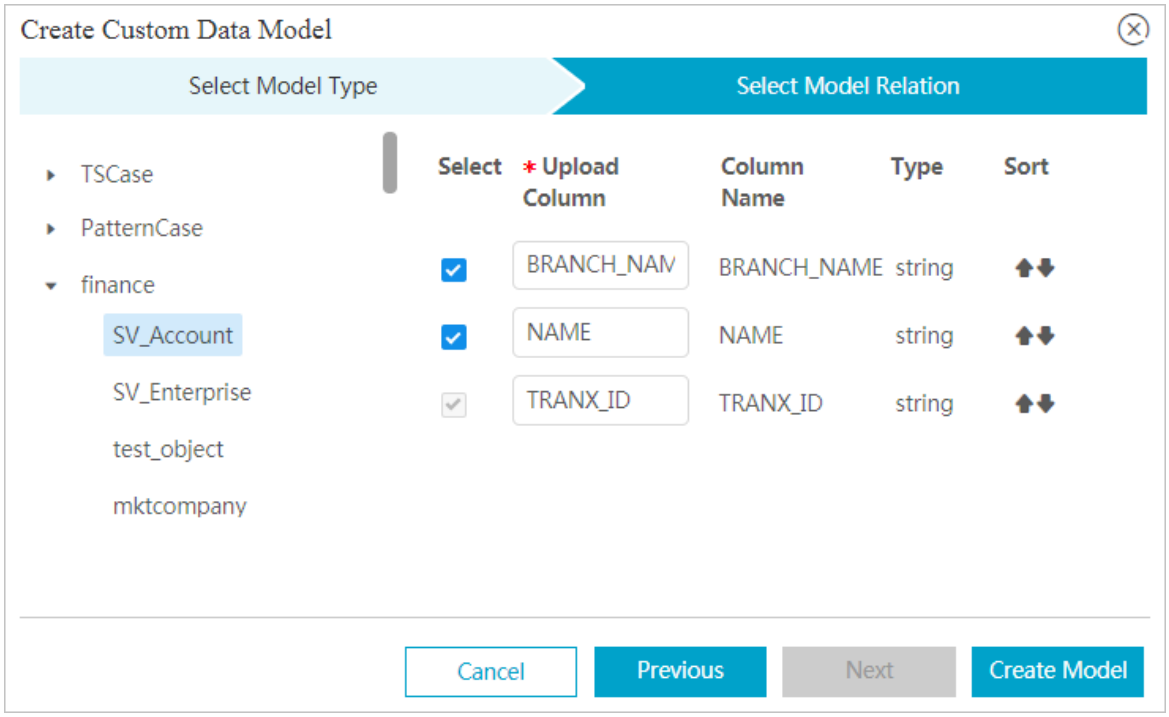


3. On the **Import Models** page, click **Create Model** in the upper-right corner of the page.
4. In the **Create Custom Data Model** dialog box, specify the **Model Type** and the **Model Name**.



5. Click **Next**.
6. Set the model relation in the **Create Custom Data Model** dialog box.

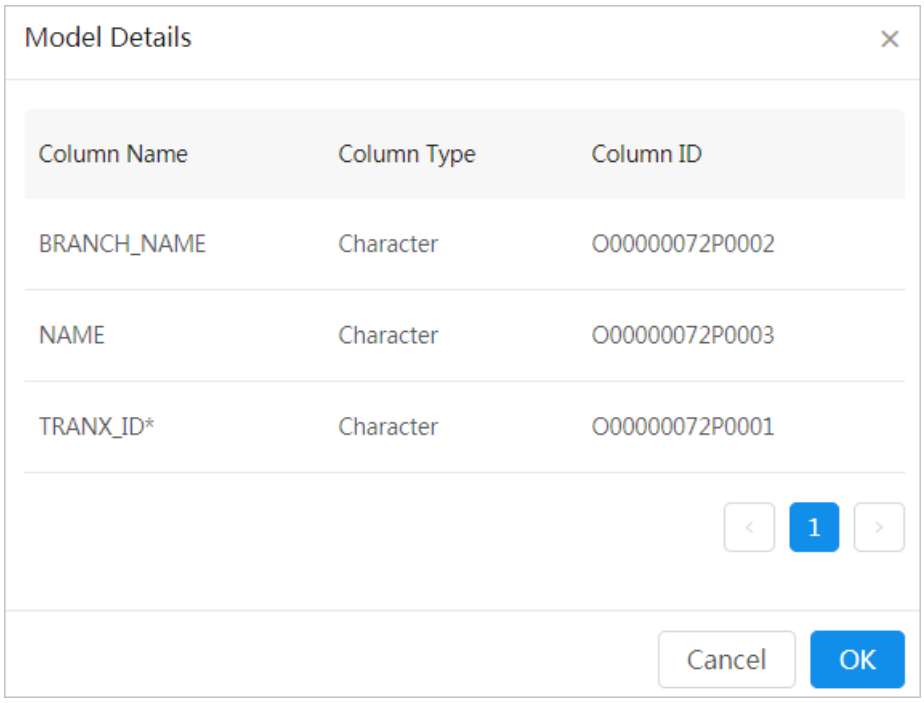
   Select an object, link, or event on the left side, and select the model columns on the right side based on the data to be imported. The columns of the primary key type are selected by default and cannot be operated.



7. After you have configured the preceding parameters, click **Create Model**.

View a model

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Import Models**.
3. On the **Import Models** page, select a model and click **View** to view the details of the model in the **Model Details** dialog box that appears.



Delete a model

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Import Models**.
3. On the **Import Models** page, select a model and click **Delete**.
4. In the dialog box that appears, click **OK** to delete the model.

8.9.2. Configure a search item

You can configure search items to set the fields to be searched for in Analytics Workbench. After a search item is configured, it must be correlated with an object, link, or event property to take effect.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- Before you delete a search item, you must first delete the links between the search item and the correlated properties of objects, links, and events.

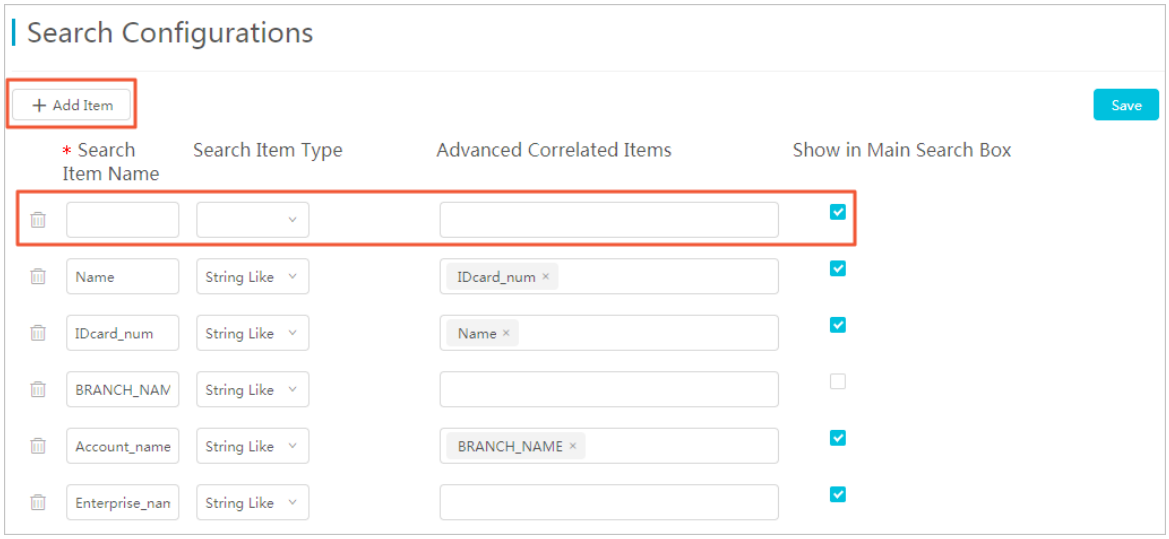
Context

This topic describes how to add, modify, and delete search items, and how to associate search items with the properties of objects, links, or events.

Add a search item

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Search Configurations**.
3. Click **Add Item** in the upper-left corner of the page that appears.

   The parameters are described in Search item configuration parameters.



Search item configuration parameters

Parameter	Description

Parameter	Description
Search Item Name	**Search Item Name** is customized by the user. We recommend that you set the name according to the properties of the objects, links, or events that need to be correlated. For example, a mobile phone number, an ID number, or a person's name.
Search Item Type	The **Search Item Type** is used to set the data type that is supported by this search item. The **Search Item Type** must be consistent with the **Query Type** of the property of the object, link, or event to be correlated with.
Advanced Correlated Items	**Advanced Correlated Items** are used to group multiple search terms to search for data. You can select multiple configured search terms.
Show in Main Search Box	Sets whether the search item is displayed in the **Search** page of Analytics Workbench.

4. (Optional)Click **Add Item** to add multiple search items as needed.
5. Click **Save**.

After a search item is added, you must correlate the search item to the properties of an object, link, or event. For more information, see Correlate a search item to properties.

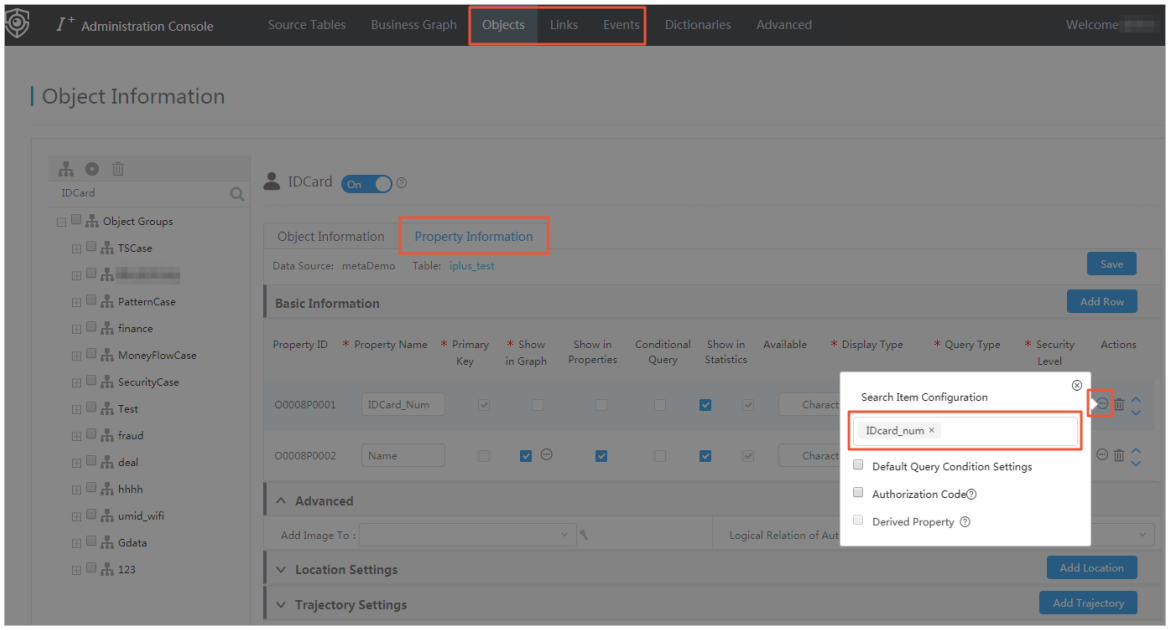
Correlate a search item to properties

The configuration entries are as follows:

- Objects: Administration Console > Objects > selected the object to be correlated to > Property Information > Basic Information > more configurations.
- Links: Administration Console > Links > select the link to be correlated to > Property Information > Basic Information > more configurations.
- Events: Administration Console > Events > select the event to be correlated to > Property Information > Basic Information > more configurations.

This topic describes how to correlate a search item with the properties of an object as an example. You can correlate a search item with the properties of a link or event by using similar methods.

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, click **Objects**.
3. In the left-side navigation pane of the **Object Information** page, click the name of the object to be configured and then click the **Property Information** tab on the right side.
4. In the **Basic Information** area, click the More icon (⊖) next to the property to be correlated, and then select the configured search items in **Search Item Configuration**.



5. After you have configured these parameters, click the icon to close the configuration box.
6. Click **Save** to save the configurations.

Modify a search item

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Search Configurations**.
3. Modify the parameters of a search item as needed.
4. Click **Save**.

Delete a search item

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **Search Configurations**.
3. Click the icon to delete the corresponding search item.
4. Click **Save**.

8.9.3. System settings

8.9.3.1. Configure components

You can configure the functional components to enable or disable functions in Analytics Workbench and Administration Console and set basic information of Analytics Workbench. When a functional component is disabled, this function will not be displayed in the operation area.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Context

The functional component settings take effect globally. Enable or disable components as needed with caution.

Procedure

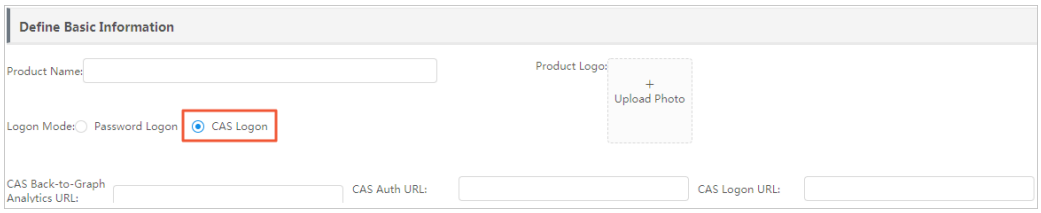
1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced > System Settings > Functional Components**.
3. Define basic information: Set the **Product Name**, **Product Logo**, and **Logon Mode** parameters for Graph Analytics.

> **Note** If you do not set the **Product Name** parameter and the **Product Logo** parameter, the default name and default Logo image are used.

If the CAS (Central Authentication Server) server is deployed in your environment, you can select **CAS logon** as the logon mode, as shown in CAS logon.

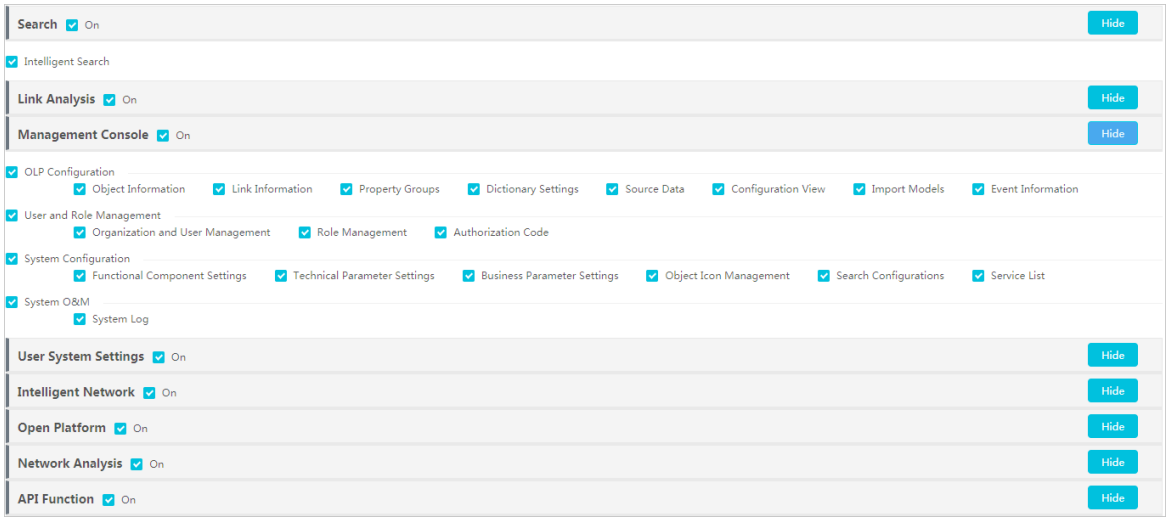
> **Note** CAS is a single sign-on protocol. If you select **CAS logon** as the logon mode, you only need to log on once to access all trusted application systems.

CAS logon



4. Enable and disable functional components: Select the modules to be enabled and cancel the modules to be disabled.

   If you select or clear a parent component, the child components will be automatically selected or cleared.



5. Click **Save**.

8.9.3.2. Technical parameters

8.9.3.2.1. Path analysis settings

In Graph Analytics, you can set the highest link degree of path analysis. Also, you can set whether to calculate only the nodes that are of the same type as the selected nodes.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Context

Typically, the default values are used for path analysis settings. Exercise caution when you modify the settings.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced > System Settings > Technical Parameters**.
3. In the **Path Analysis Settings** area, you can set the parameters based on your requirements.

   The parameters are described in Path analysis configuration parameters.

Path analysis configuration parameters

Parameter	Description

Parameter	Description
Current Path Analysis Degree	The highest link degree supported by path analysis. The default value is 2 and the maximum value is 3. If the **Current Path Analysis Degree** parameter is set to *N*, only links of degree *N* or lower degrees will be analyzed. Links of a degree higher than *N* will not be analyzed.
Restrict Node Types in Path Analysis	Valid values: ○ **Yes**: Only nodes of the same type as the selected node are calculated. ○ **No**: All types of nodes are calculated. The default value is **No**.

4. Click **Save** in the upper-right corner.

8.9.3.2.2. Quick extension settings

Graph Analytics, you can set the highest link degree for quick extension.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Context

Typically, the default values are used for quick extension settings. Exercise caution when you modify the settings.

> **Note** If the degree is set too high, the running performance of the system will be affected.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Technical Parameters**.
3. In the **Quick Extension Settings** area, you can set the **Extension Degree** parameter based on your requirements.

Extended Degree: The highest link degree for quick extension. The default value is 2. If the **Extended Degree** parameter is set to *N*, only links of degree *N* or lower degrees will be analyzed. Links of a degree higher than *N* will not be analyzed.

4. Click **Save** in the upper-right corner.

8.9.3.2.3. Maximum node settings

In Graph Analytics, you can set the maximum number of nodes to be queried at the same time when you perform an analysis.

Prerequisites

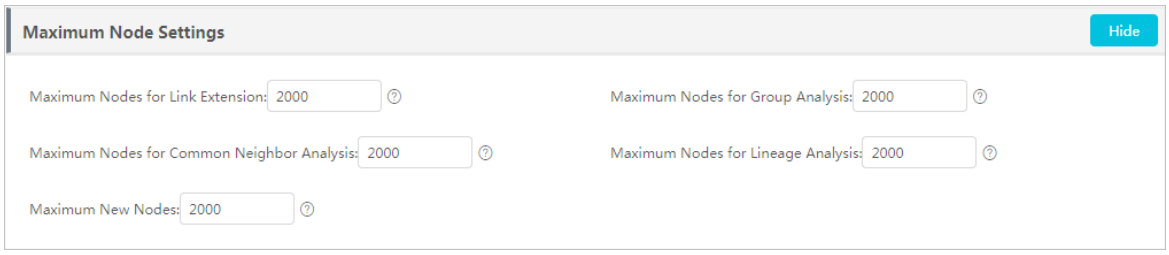
Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Context

Typically, the default value is used for the maximum number of nodes to be queried. Exercise caution when you modify the number of nodes.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Technical Parameters**.
3. In the **Maximum Node Settings** area, you can configure the parameters based on your requirements.



4. After you have completed the configurations, click **Save** in the upper-right corner.

8.9.3.3. Business parameters

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **System Settings** > **Business Parameters** from the top navigation bar. The Business Parameters page appears.

8.9.3.3.1. Add double-click link settings

In Analytics Workbench, you can double-click an object to query the relationship between objects. In Administration Console, you can custom the relationships of an object to be queried when you double-click the object. If no custom configuration is set, all the first-degree relationships of the object are queried by default when you double-click the object.

Prerequisites

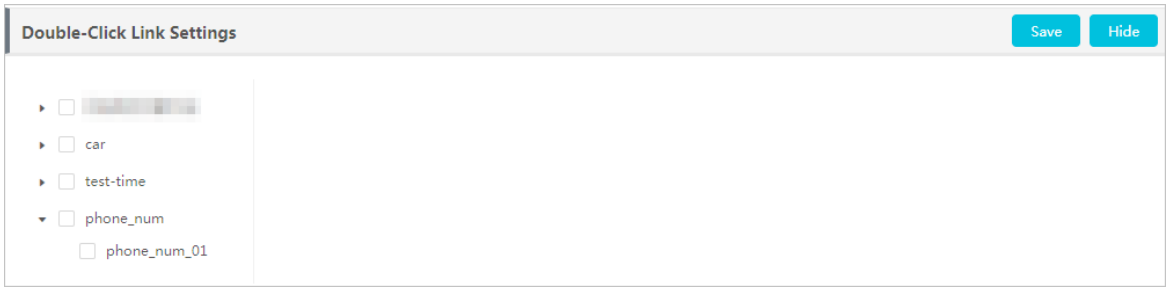
- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- The object has been created and referenced by the link.

Context

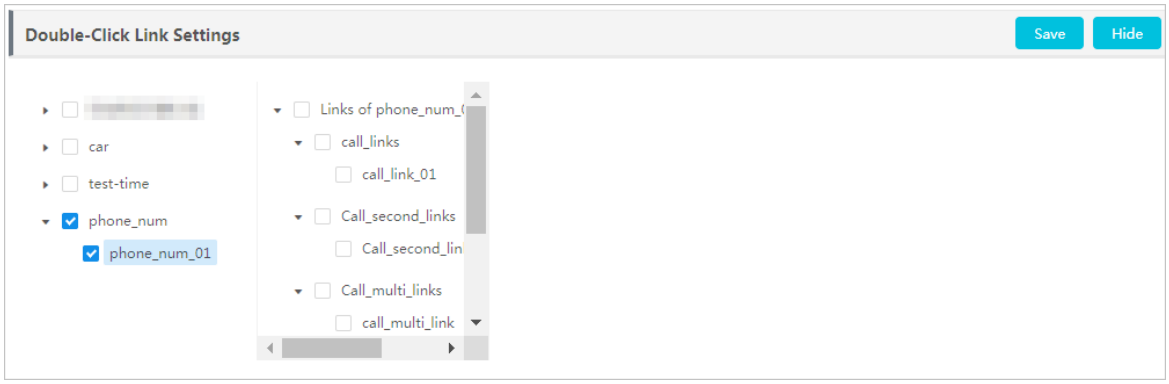
The double-click link settings take effect globally. Exercise caution when you configure the settings.

Procedure

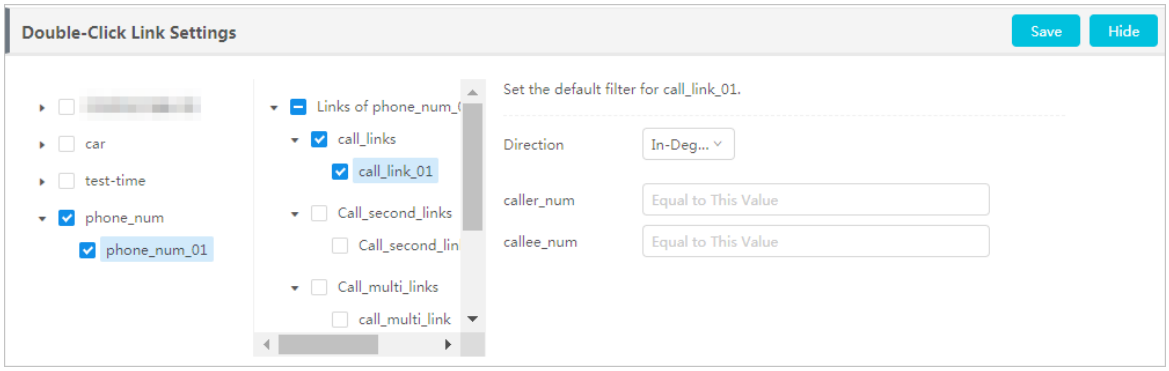
1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Business Parameters**.
3. In the **Double-Click Link Settings** area, select the objects to be configured in the left-side list.



4. In the list that is displayed on the right side of the area, select the links to be queried.



5. In the displayed area, set the filter conditions for querying the link.



6. (Optional)If you need to query other links by double-clicking the object, you can continue to select other links and set the filter conditions.
7. Click **Save** to complete the double-click link settings of the object.

8.9.3.3.2. Double-click-disabled object settings

In Analytics Workbench, you can double-click a node to query the relationship network of the node. Administration Console allows you to enable or disable double-clicking on a specified object. By default, double-clicking is supported for all objects.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created an object.

Context

The double-click-disabled object settings take effect globally. Exercise caution when you configure the settings. We recommend that you disable the double-click operations on objects that generate large amounts of data.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced > System Settings > Business Parameters**.
3. In the **Double-Click-Disabled Object Settings** area, click the **Double-Click-Disabled Objects** drop-down list, and select the objects.

4. Click **Save**.

8.9.3.3.3. Object grouping settings

Graph Analytics allows you to set the grouping conditions, so that objects or links of the same type can be grouped automatically when their quantity reaches the threshold value. In a complex graph analytics, we recommend that you set reasonable grouping conditions to keep the analysis graph concise and clear.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created an object.

Context

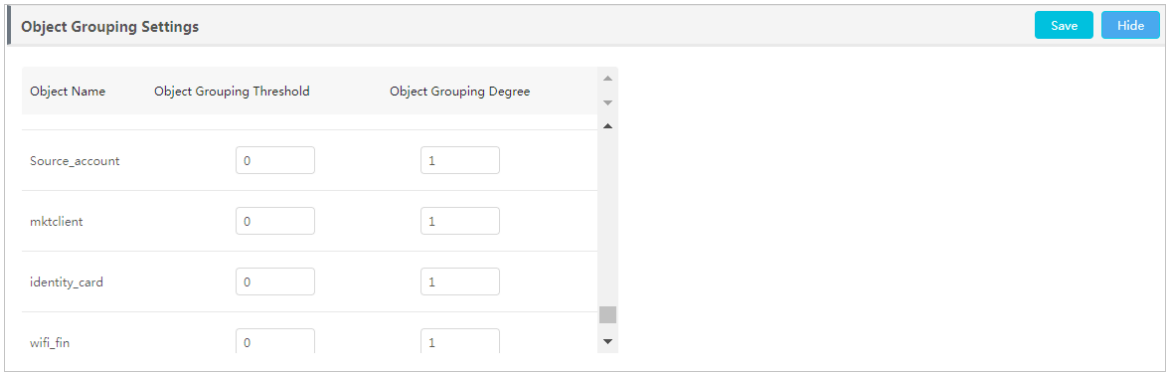
The **Object Grouping Threshold** and **Object Grouping Degree** parameters are described as follows:

- **Object Grouping Threshold**: If the number of object nodes of the same type that are connected to the same node exceeds the threshold, the connected nodes are grouped into one folder.
- **Object Grouping Degree**: If more than one node has the link of the specified degree with the same nodes, these nodes will be grouped into a folder. For example, set **Object Grouping Degree** to 2. If both node A and node B have second-degree links with nodes C and D, node A and node B will be grouped into one folder.

The value of **Object Grouping Threshold** defaults to 0, and the value of **Object Grouping Degree** defaults to 1. The default value indicates that no objects will be grouped automatically.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Business Parameters**.
3. In the **Object Grouping Settings** area, specify **Object Grouping Threshold** and **Object Grouping Degree** based on your requirements.



4. Click **Save**.

8.9.3.3.4. Configure lineage analysis

Before you perform a lineage analysis on a specified object node in Analytics Workbench, you need to configure the business parameters for this type of object in Administration Console.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- The object has been created and referenced by a link.

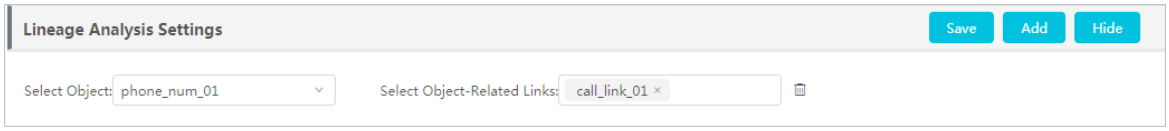
Context

Lineage analysis extends a specific business link to multiple degrees. The specific link typically refers to the lineage link and the same residence number.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Business Parameters**.
3. In the **Lineage Analysis Settings** area, click **Add** to add a new setting.

4. Select the object to be analyzed based on your requirements, and then select the object-related link to be analyzed.



5. Click **Save**.

8.9.3.3.5. Intimacy measurement settings

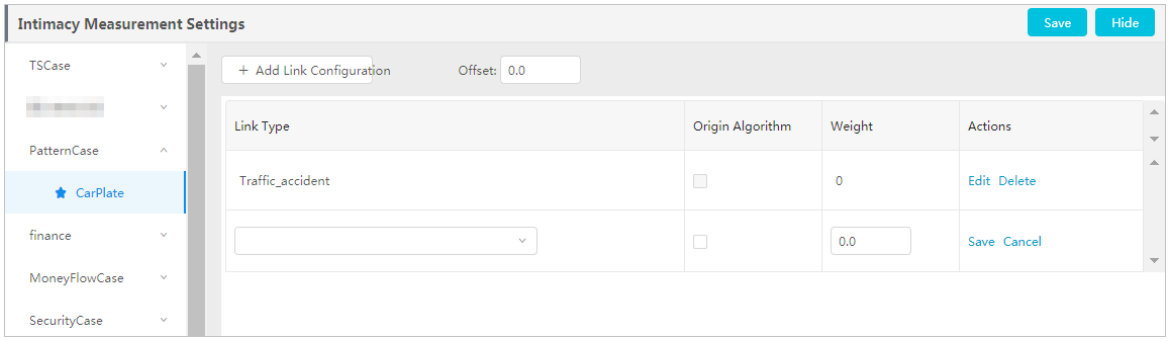
You can set the weight for the relationship between specified objects, so that you can directly view the intimacy between these objects in the analysis result.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- An object has been created and has been referenced by a link.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Business Parameters**.
3. On the left side of the **Intimacy Measurement Settings** area, select an object and then click **Add Link Configuration** on the right side of the area.
4. Select a link type in the **Link Type** drop-down list, and then set the intimacy weight and other parameters.



The star icon in front of an object indicates that the object has configured intimacy measurement settings, as shown in **ID Card** in the figure above.

5. After you have configured these parameters, click **Save**.
6. (Optional)To add intimacy measurement settings for another **Link Type**, click **Add Link Configuration**.

   To add intimacy measurement settings for another object, select the object in the left-side navigation pane, and then click **Add Link Configuration**.
7. After you have configured the intimacy measurement settings for all objects, click **Save** on the right side of the **Intimacy Measurement Settings** section.

8.9.3.3.6. Redirect URL settings

When you need to redirect to an external system from Graph Analytics, you can configure the URLs in External System Redirect URL Settings.

8.9.3.4. Object icons

8.9.3.4.1. Upload an object icon

You can upload a local object avatar to the icon library in Graph Analytics.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- Make sure that you have prepared an avatar image in the PNG format. The recommended size is 32px * 32px. We recommend that you limit the size to 320px * 320px.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Object Icons**.
3. On the **Object Icons** page, click **Upload Icon**.
4. In the **Upload Icon** dialog box that appears, click the upload area to upload a local icon, and then specify the **Name** column.
5. Click **OK** to upload the object icon.

8.9.3.4.2. Modify an object icon

You can modify the avatar image or the avatar name based on your requirements.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- Make sure that you have prepared an avatar image in the PNG format. The recommended size is 32px * 32px. We recommend that you limit the size to 320px * 320px.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top menu bar, choose **Advanced** > **System Settings** > **Object Icons**.
3. On the **Object Icons** page, move your mouse pointer over the icon, and click the Modify icon ( ).
4. In the **Edit** dialog box that appears, modify **Name** or upload a new avatar image.
5. Click **OK**.

8.9.3.4.3. Delete an object icon

You can delete the object icons that are no longer used.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- Before you delete an object icon, make sure that this icon is not used by another object.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. In the top navigation bar, choose **Advanced** > **System Settings** > **Object Icons**.
3. On the **Object Icons** page, move your mouse pointer over the icon, and click the Delete icon ( ).
4. In the dialog box that appears, click **OK** to delete the object icon.

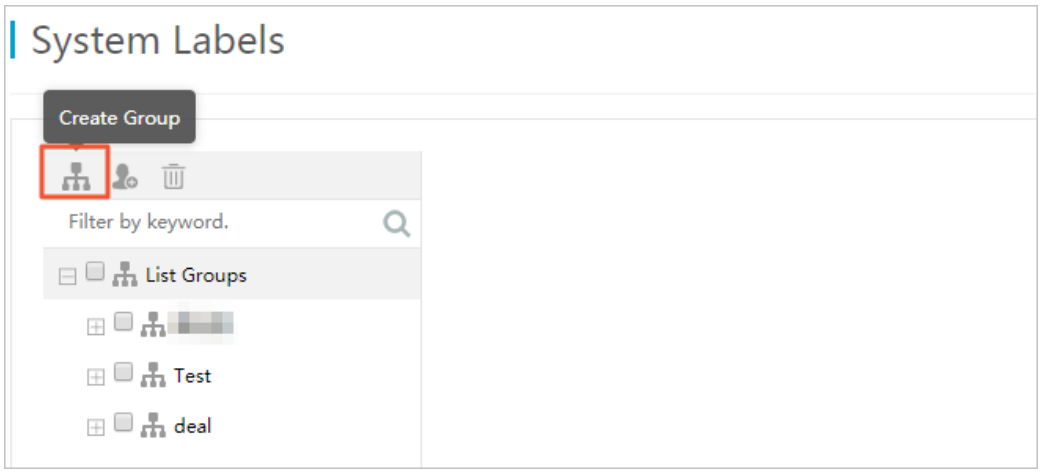
8.9.4. System labels

8.9.4.1. Create a group

Procedure

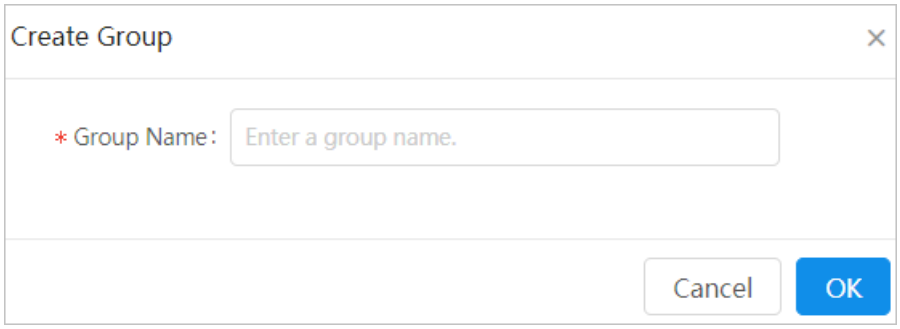
1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **Users and Permissions** > **System Labels** from the top navigation bar.
3. On the **System Labels** page, click the **Create Group** icon, as shown in Create a group.

   Create a group



4. In the Create Group dialog box that appears, enter a group name, as shown in Enter a group name.

   Enter a group name



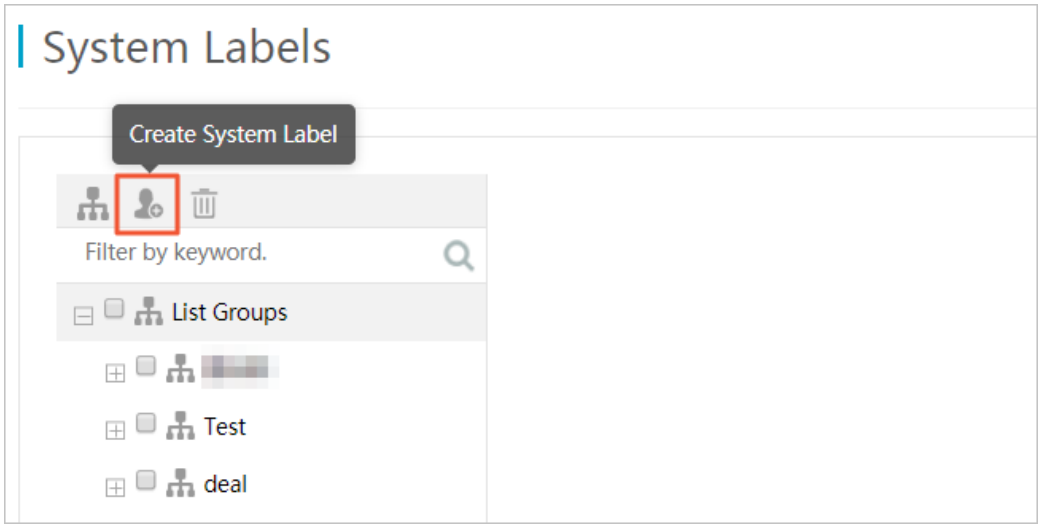
5. Click **OK**.

8.9.4.2. Create a system label

Procedure

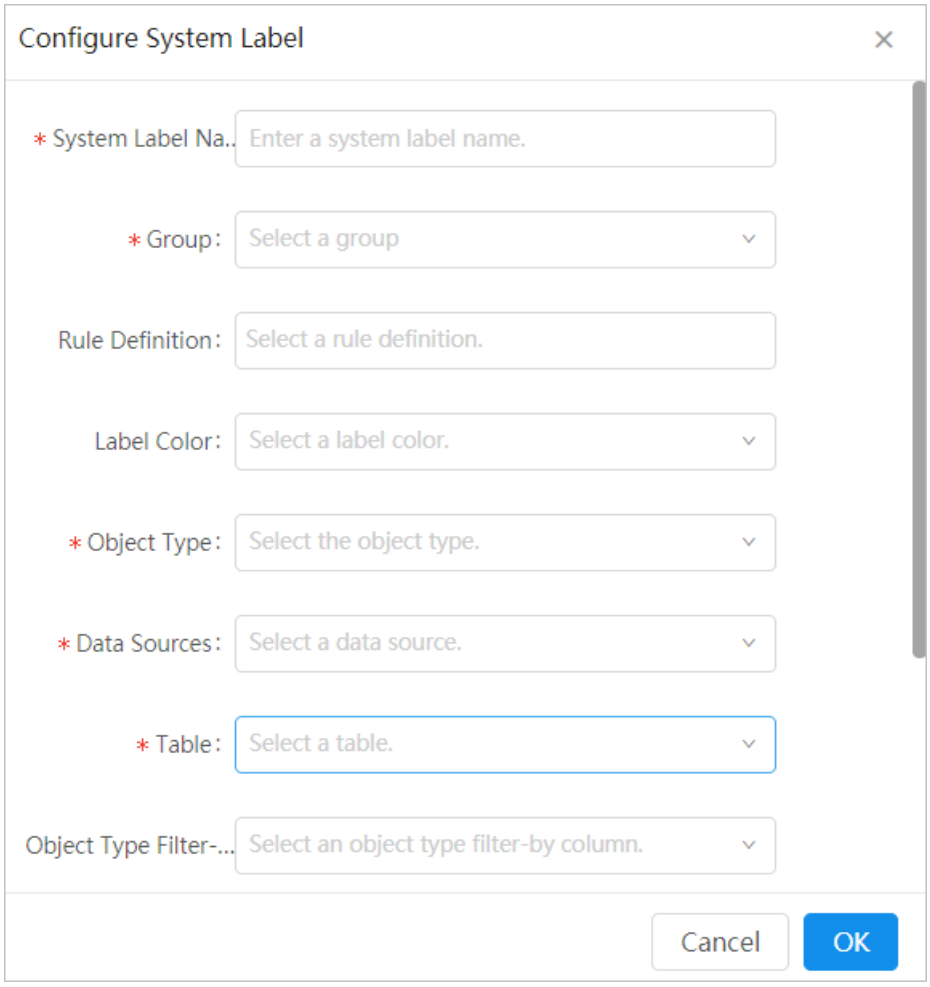
1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **Users and Permissions** > **System Labels** from the top navigation bar.
3. On the System Labels page, click the **Create System Label** icon, as shown in Create a system label.

   Create a system label



4. In the **Configure System Label** dialog box, configure the required parameters, as shown in Configure a system label.

   Configure a system label



The system label parameters are described as follows:

- System Label Name: the alias of the system label.
- Group: the group to which the system label belongs.
- Rule Definition: The processing rule that is applied when an object matches the system label. Value options include Not Display Object, Not Extend Object, Not Display Properties, Disable Property Statistics, and Disable Behavior Display. If you have selected Not Display Object, the other four options and the Label Color parameter become unavailable.
- Label Color: the display color of the system label.
- Object Type: all available objects are displayed in the drop-down list. Choose an object as needed.

  After you select the object type, a primary key parameter will appear in the dialog box. Select a column corresponding to the primary key property as needed.
- Data Source: the data source where the system label belongs.
- Table: the table where the system label belongs.
- Object Type Filter by Column, Object Type Filter Operator, and Object Type Filter Value: Optional. You can use these parameters to filter system labels. If you have specified any one of these parameters, the other two parameters are required.

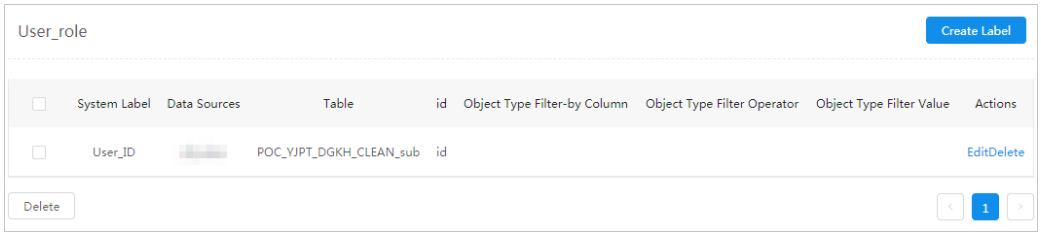
5. Click **OK**.

8.9.4.3. Modify a system label

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **Users and Permissions** > **System Labels** from the top navigation bar.
3. On the System Labels page, click an object from the left-side navigation pane.
4. In the details area on the right side of the page, select a system label, and click **Edit**, as shown in Modify a system label.

   Modify a system label



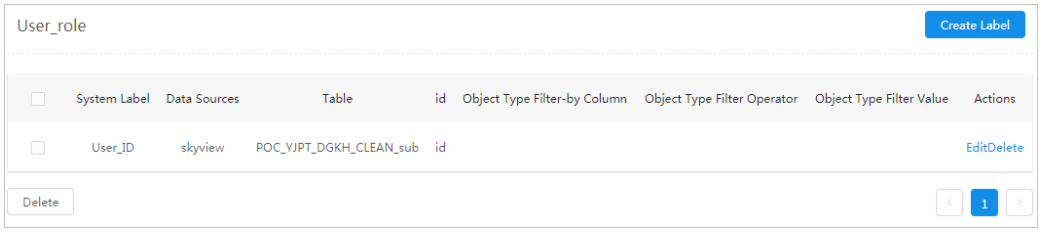
5. Modify the parameters as needed. The object type cannot be modified.
6. Click **OK**.

8.9.4.4. Delete a system label

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **Users and Permissions** > **System Labels** from the top navigation bar.
3. Click **Delete** next to a system label, or select one or more system labels and then click **Delete** in the lower-left corner, as shown in Delete a system label.

   Delete a system label



8.9.5. System operations and maintenance

8.9.5.1. Audit logs

Procedure

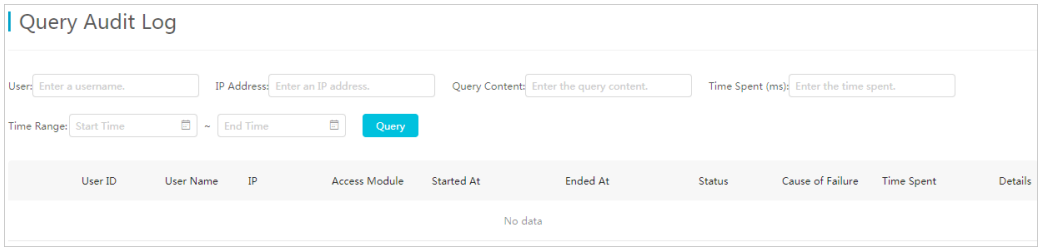
1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **System O&M** > **Audit Log** from the top navigation bar. The Query Audit

Log page appears.

3. Set the search filters, and then click **Query**, as shown in Log query.

   If you click **Query** without setting any filters, all data will be returned.

   Log query



8.9.6. View server clusters

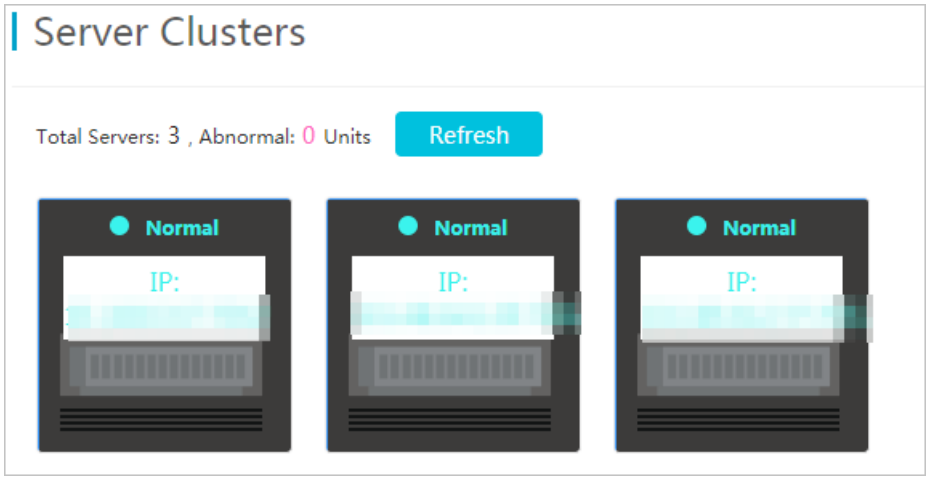
In Administration Console, you can view the information of all servers in a cluster, including the running status, server exceptions, the number of servers, IP addresses, and port numbers.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Administration Console of Graph Analytics.
2. Choose **Advanced** > **Server Clusters** to go to the **Server Clusters** page.



   In Administration Console, you can view the information of all servers in a cluster, including the running status, the number of servers and server exceptions, IP addresses, and port numbers.

3. If you have stayed on the current page for too long, you can click **Refresh** to view the latest information.

8.10. Import data

8.10.1. Model list

8.10.1.1. Model overview

A model is a template used to import data to Graph Analytics. You can use models to import individual or small amounts of data to Graph Analytics.

Models include custom models and system models:

- Custom models are directly created by users on Analytics Workbench. Only the creator can view, download, modify, and delete the custom models.
- System models are created by the administrator in Administration Console. All users can view and download system models on Analytics Workbench. However, system models cannot be modified or deleted. To delete a system model, you must log on to Administration Console.

8.10.1.2. View models

In Analytics Workbench, you can view information about the existing data models, including the model type, mapped OLEP, properties, and the property type. This helps you understand the existing models at any time and identify the data model that matches the data to be imported.

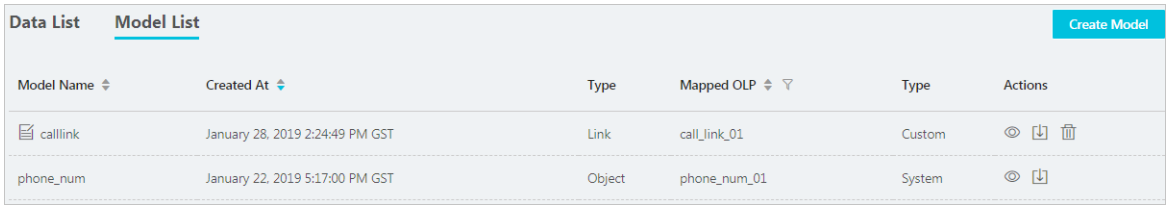
Prerequisites

All users can view the system models. However, to view a custom model, you must have the account and password of the user that created the model.

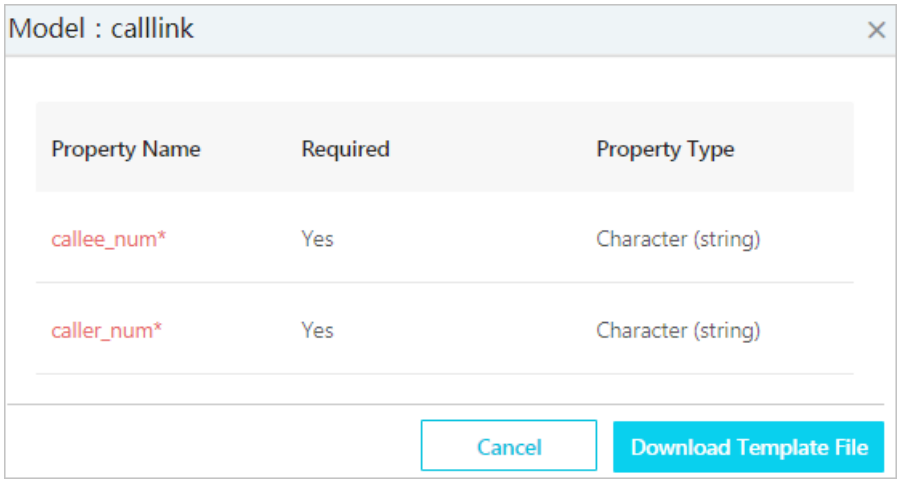
Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page, and click the **Model List** tab. On the **Model List** page, you can view the creation time, model type, and mapped OLEP for all models.

   > **Note** Analytics Workbench supports custom and system models. Custom models are created by users on Analytics Workbench. For more information, see Create models. System models are configured in Administration Console. For more information, see Manage a system model.

3. Click the icon next to the data model that you want to view, and view the property information about the data model in the dialog box that appears.



If the data model meets the requirements for your data import, you can download this model and use this model as a template to sort the data to be imported.

4. (Optional)Click **Download Template File** to download the model file in the XLSX format.

8.10.1.3. Create models

The model list defines a column format for the uploaded file. The columns of the uploaded file will be mapped to the corresponding properties, including the object property, link property, object and link property, and event property. If the existing models cannot meet the requirements of the data to be imported, you must create a data model based on the target data.

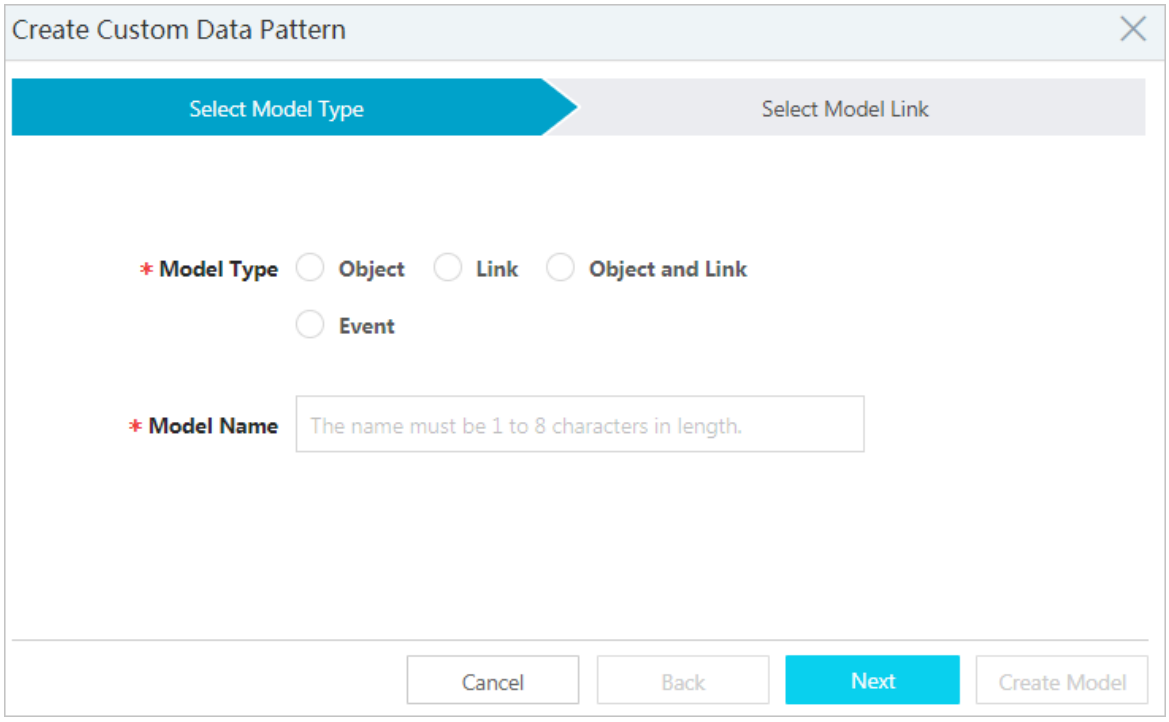
Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page, and then click the **Model List** tab. The **Model List** page appears.
3. Click **Create Model** in the upper-right corner of the page, and set the parameters in the **Create Custom Data Model** dialog box that appears.

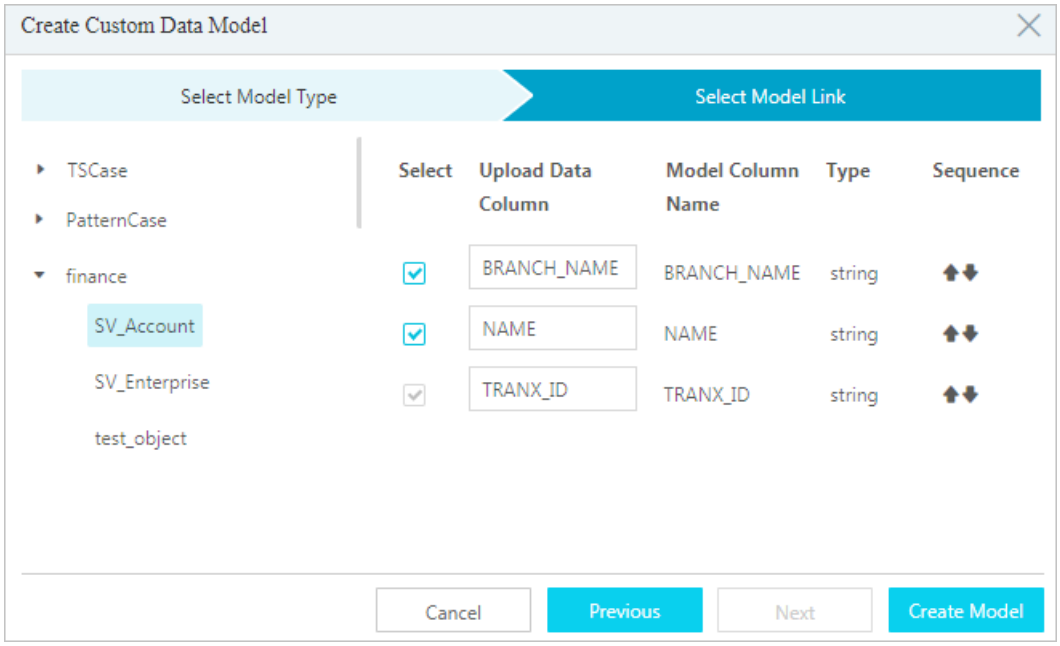
   For more information about the parameter settings, see Parameter configurations.



Parameter configurations

Parameter	Description
Model Type	You can create multiple model types, including **Object**, **Link**, **Object and Link**, and **Event**. Specify a model type as needed.
Model Name	Enter a model name. We recommend that you enter a name that is easy to understand.

4. Click **Next**, select an object, link, or event, select the required columns, and then set the name of **Upload Data Column**.



5. Click **Create Model**. A success message is displayed, indicating that the model has been saved.

8.10.1.4. Modify model names

If a model name is obscure or does not match the model content, you can modify the model name.

Prerequisites

Make sure that you have obtained the account and password of the user that created the model that you want to modify.

> **Note** In Graph Analytics, you can only modify the names of user-defined models. You are not allowed to modify the system models.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page, and click the **Model List** tab. The **Model List** page appears.
3. Click the icon in the front of the model that you want to edit, rename the model, and then click **OK**.

8.10.1.5. Download a model

Before you import data, you must download a compatible model, which is an .xlsx file, and use this model as a template to sort the data to be imported.

Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page, and click the **Model List** tab. On the **Model List** page, you can view the creation time, model type, and mapped OLEP for all models.
3. Click the icon next to the model that you want to download, and save the model as prompted.

What's next

Use this model as a template to organize the data to be imported, and then import the collected data to Graph Analytics. For more information about the detailed operations, see Import data.

8.10.1.6. Delete a model

You can delete the custom models that are no longer used.

Prerequisites

Make sure that you have obtained the account and password of the user that created the model you want to delete.

> Note Graph Analytics only supports deleting user-defined models. To delete a system model, you need to log on to Administration Console. For more information, see Manage a system model.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page, and click the **Model List** tab. On the **Model List** page, you can view the creation time, model type, and mapped OLEP for all models.
3. Click the icon next to the model that you want to delete.

8.10.2. Import data

Analytics Workbench supports importing data in the csv, txt, xls, or xlsx format. You can analyze small amounts of data or individual pieces of data that are missing from the data source.

Prerequisites

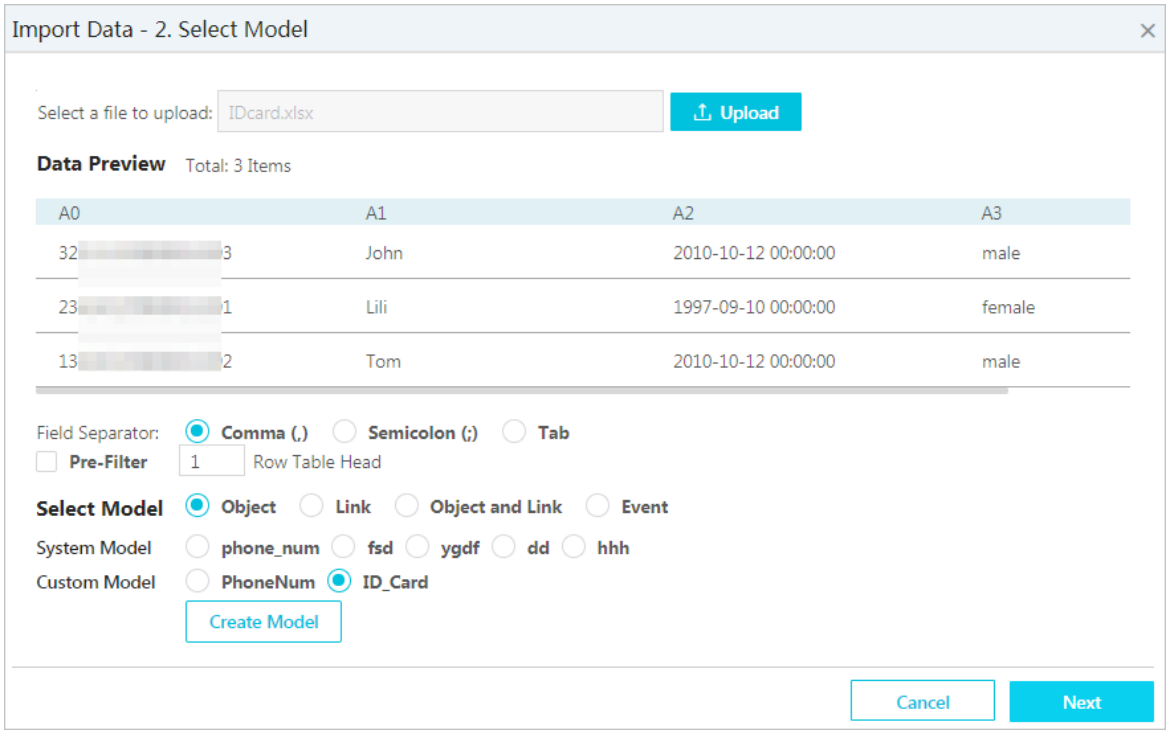
- The import data source is added when you configure the data source Create data sources.
- You have obtained an account and a password with the **Import Data** permissions.
- You have created a model matched to the data to be imported. For more information, see Create models.
- You have organized the data to be imported according to the template file, or sorted the data in the format required by the template, such as the csv, txt, xls, or xlsx format.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner. In the dialog box that appears, select the file to be imported.
3. Click **Upload**, and set the read format of the file according to the data content to be imported.

   The parameter configurations are described in Parameter configurations and descriptions.

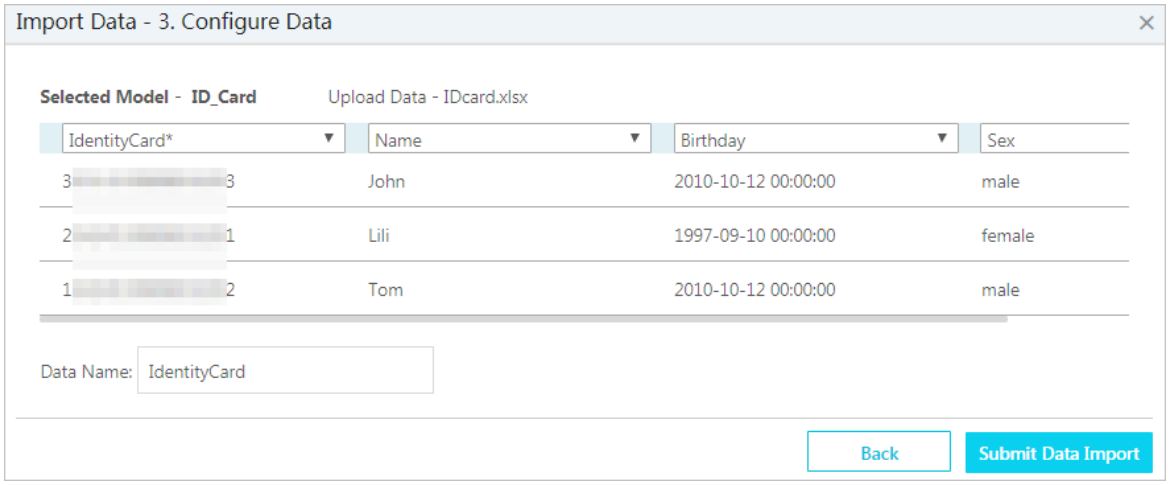
   After the file is imported, you can see the data preview of the file in the dialog box. Only the first 10 lines of the file content and the total number of lines are displayed.



Parameter configurations and descriptions

Parameter	Description
Column Separator	Sets the internal column separator for each line of the content.
Encoding Method	Specifies the character encoding for the file content.
Select Model	After you have selected the model type corresponding to the uploaded data, you can see the currently available system models in the **System Model** tab and custom models in the **Custom Model** tab. If you do not have a model that matches the data to be imported, you can click **Create Model** to create a new model in real time. For more information, see Create models.

4. After you have configured the parameters, click **Next** to set the data name and the mapping relationship between the columns and model properties of the data to be imported.



5. After you have configured the parameters, click **Submit**.

8.10.3. Data list

8.10.3.1. View data

Analytics Workbench allows you to view the imported data at any time, so that you can better understand the data.

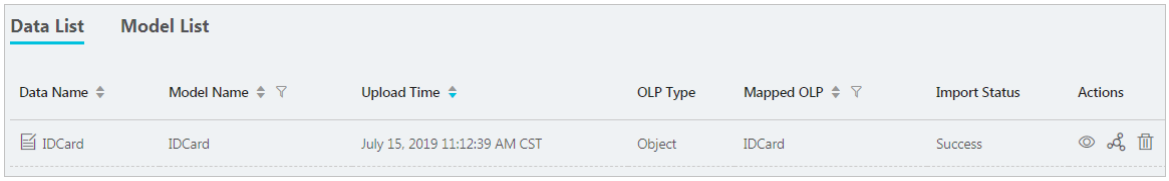
Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page to go to the **Data List** tab page. You can view all data that has been imported.

   The Data List page displays information about all imported data, including the data name, model name, upload time, OLEP type, mapped OLEP, and the import status.



3. Click the icon next to a data model that you want to view, and view the property information about this data in the dialog box that appears.

8.10.3.2. Edit a data name

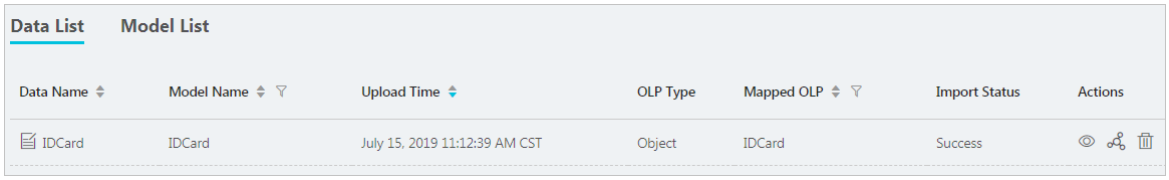
If a data name is obscure or does not match the data content, you can modify the data name.

Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page to go to the **Data List** page.



3. Click the icon in front of the data name, rename the data, and then click **OK**.

8.10.3.3. Import data to Graph

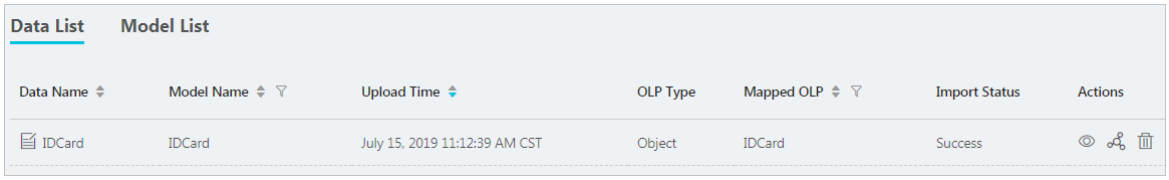
You can use this feature to import a data file to Graph to analyze the data quickly.

Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page to go to the **Data List** page.



3. Click the icon next to the specified data file to present the data in Graph.

8.10.3.4. Delete data

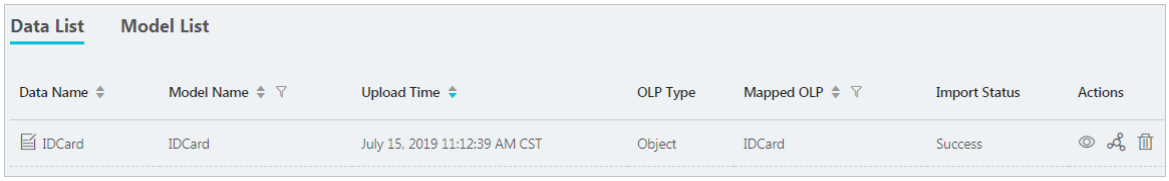
You can delete unnecessary data.

Prerequisites

You have obtained the account and password with the **Import Data** permissions.

Procedure

1. Log on to Analytics Workbench.
2. Click the icon in the upper-right corner of the page to go to the **Data List** page.



3. Click the icon next to the specified data to delete the data from Graph Analytics.

8.11. Search

8.11.1. Search

Search is one of the two key modules of Graph Analytics. Research staffs can use the Search module to find and view different objects, such as mobile phones or identity card information. In this topic, you can learn about the features and the entry of the Search interface.

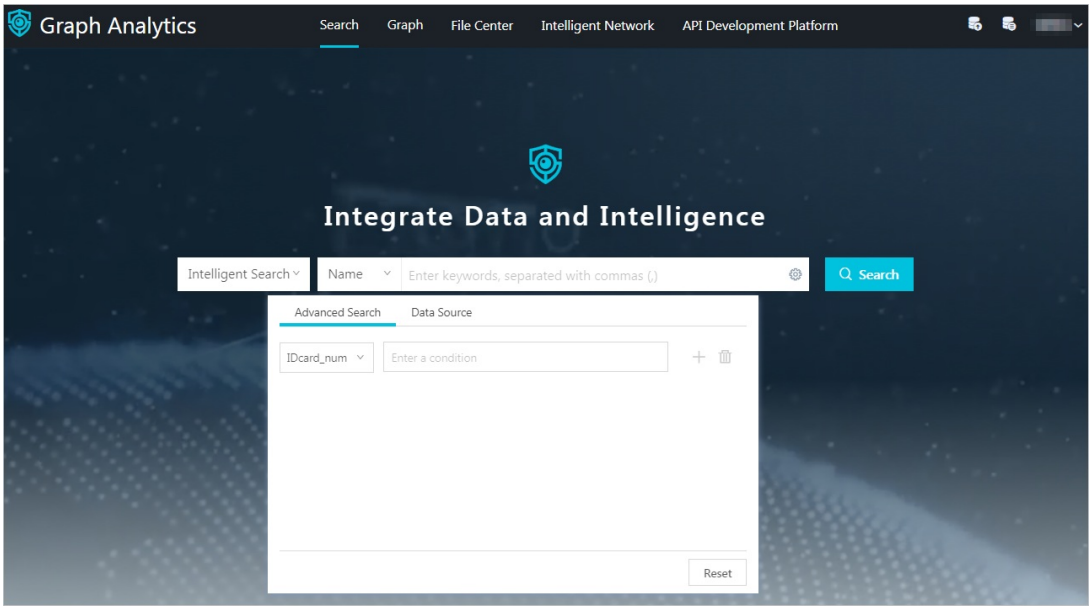
If you have obtained some fuzzy information, you can use the Search module to find objects and link records related to the information. These records can be added as independent object nodes to the relationship analysis, and can be used as a starting point for analysis and decision-making.

In the analysis process, you can expand the information step by step and refine the analysis by keywords, such as name and address. The Search module provides a search tool that allows you to retrieve the object information and locate the target information quickly.

> **Notice** In case of empty search conditions, no results will be returned, and a message Enter at least one condition value. will appear.

The Search module also serves as the entry to the Graph module. You can import the information of retrieved objects to the Graph module for further link extensions and link analyses.

Search



8.11.2. Simple search

You can use this feature to quickly search for objects that contain a certain type of keyword. Fuzzy search is supported.

Prerequisites

A search item has been configured for the target object, and the search item has been associated with the property of the object. For more information, see Configure a search item.

Context

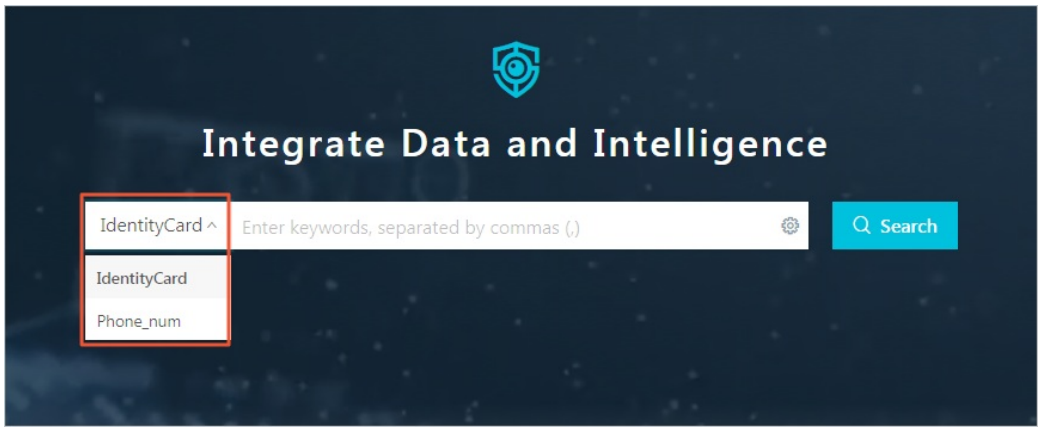
When you perform a simple search, you only need to select a keyword type and enter one or more keywords.

Procedure

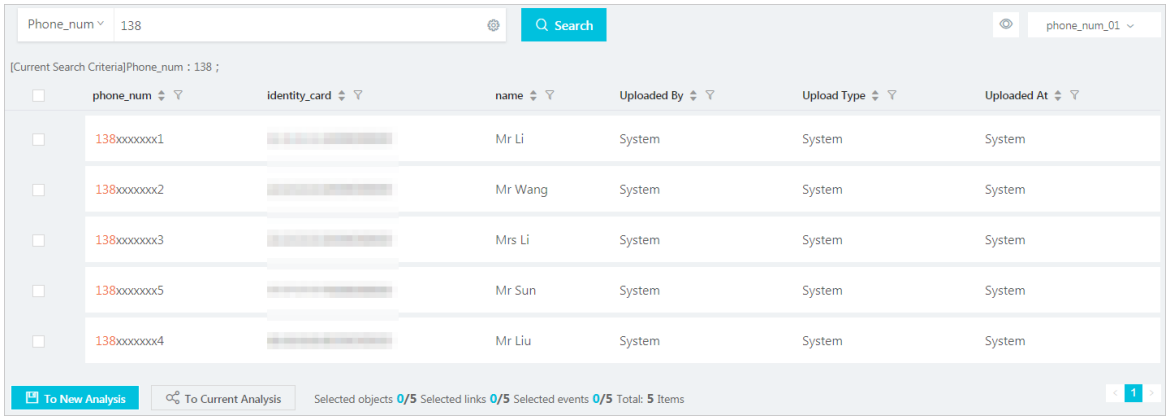
1. Log on to Analytics Workbench.
2. Click **Search** on the top navigation bar to go to the **Search** page.
3. Select a search item in the drop-down list as a keyword type, and enter one or more key words based on the search item you select.

Fuzzy search is supported. For example, if you want to search for objects whose phone number contains 138, you can just enter 138 in the search box.

Search box



4. Click **Search** or press **Enter** to start a search.



8.11.3. Advanced search

Advanced Search supports fuzzy search and multiple search conditions.

Prerequisites

You have configured a search item with an advanced association item for the target object, and the search item has been associated with the property of the object. For more information, see Configure a search item.

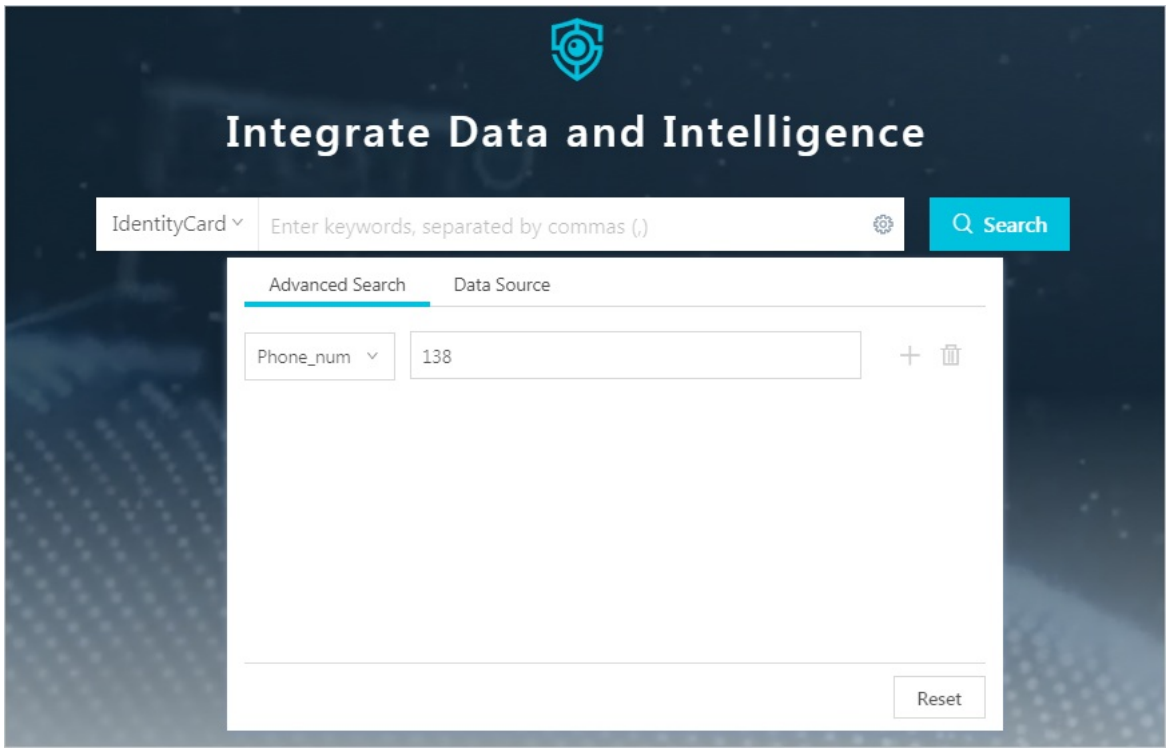
Context

You can specify the search terms in Advanced Search in the same way you perform a simple search. You can specify the advanced correlated items for the selected search terms. This is similar to a combined search based on multiple keyword types. You can also specify the data source items to be searched, which is similar to specifying the search range.

This topic describes a sample search for objects named **liqing344** whose identity card contains **234**.

Procedure

1. Log on to Analytics Workbench.
2. Click **Search** on the top navigation bar to go to the **Search** page.
3. Select a search item in the drop-down list as a keyword type, and enter one or more key words based on the search item you select. In this example, the search item is set to **ID Card** and the keyword is **234**.
4. Click the icon next to the search box to set the condition to perform an advanced search.

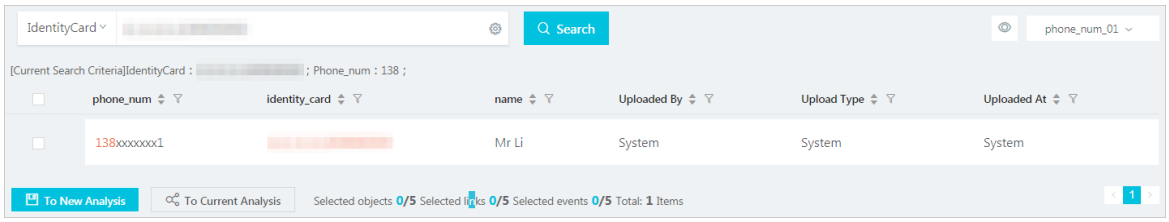


Advanced Search displays the Advanced Correlated Items of the currently selected item. In this case, the **advanced correlated item** is set as **Name**, and the keyword is **liqing344**.

Data Source: The data sources are the objects and links defined in Graph Analytics. Here you only need to select the data sources you want to search.

Reset: You can reset the search condition to the initial status.

5. After you have configured these parameters, click **Search** or press **Enter** to start a search.



8.11.4. View and analyze search results

After you have completed a search, you can view the search results and send the specified content to the graph to perform an analysis.

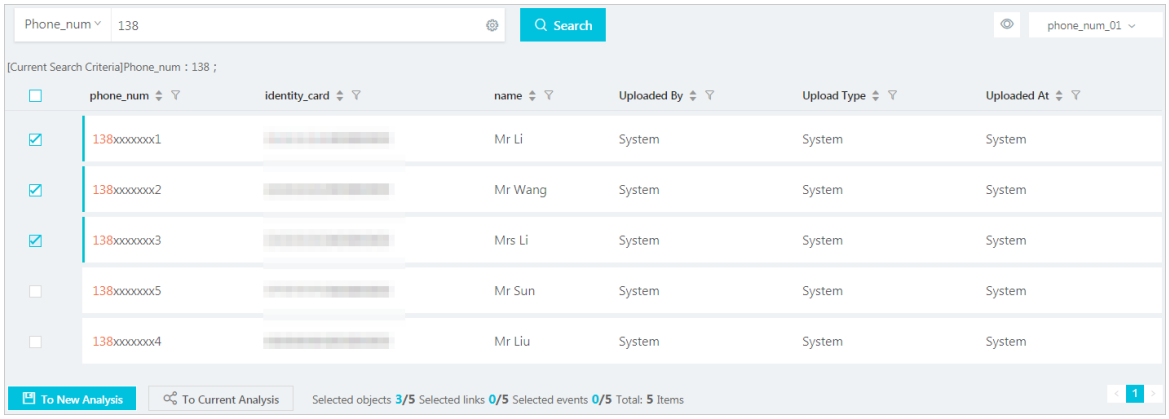
Prerequisites

You have completed a search, and the search results are not empty. For more information, see Simple search or Advanced search.

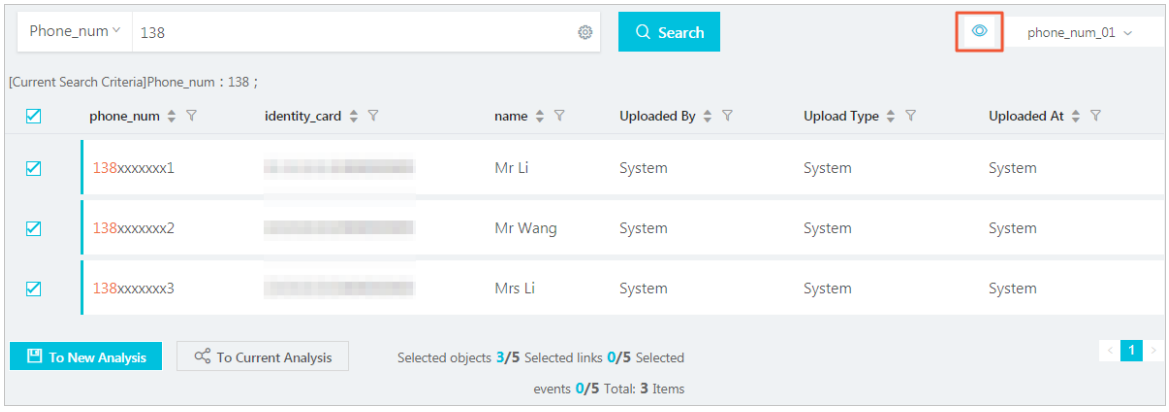
Procedure

1. Log on to Analytics Workbench.
2. Search for objects and view the search results. For more information, see Simple search or Advanced search.

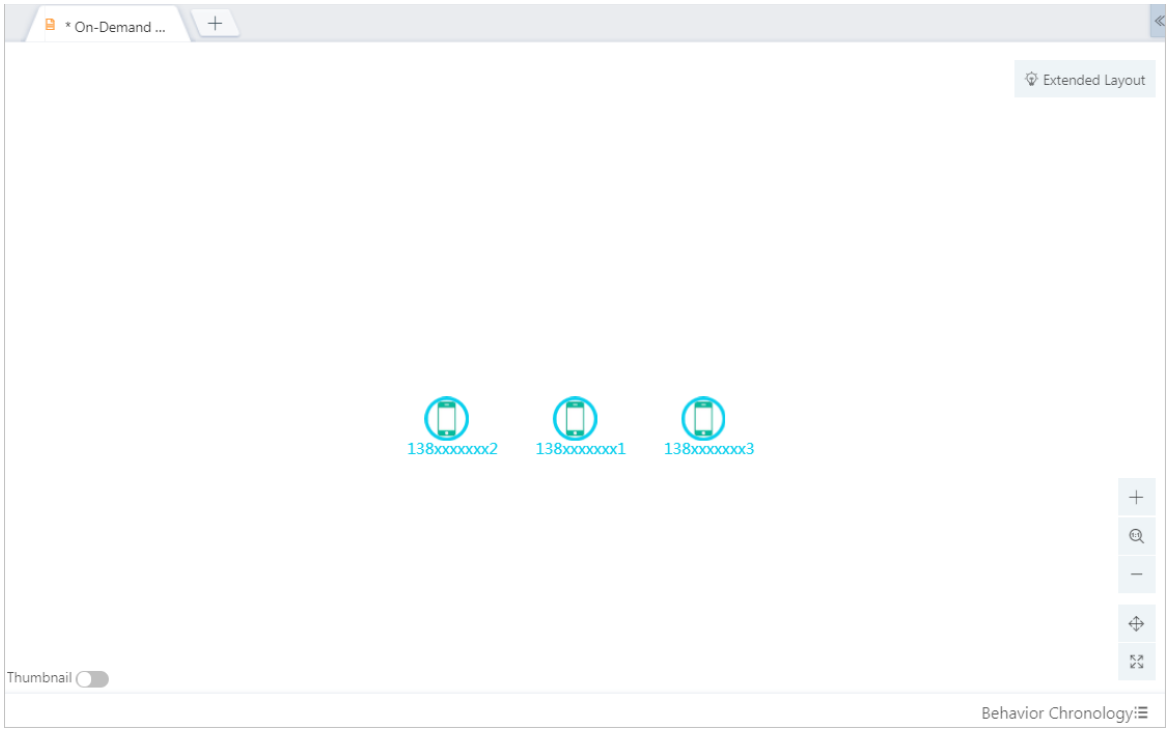
   The search results for this example are as follows:



3. Select part of or all of the search results. A total of 10 records are displayed on the current page. Click the icon in the upper-right corner. The search results only show the information of the selected objects.



4. Click the **To New Analysis** icon in the lower-left corner, or the **To Current Analysis** icon. Send the selected search results to the graph to perform an analysis.



8.12. Graph

8.12.1. Graph

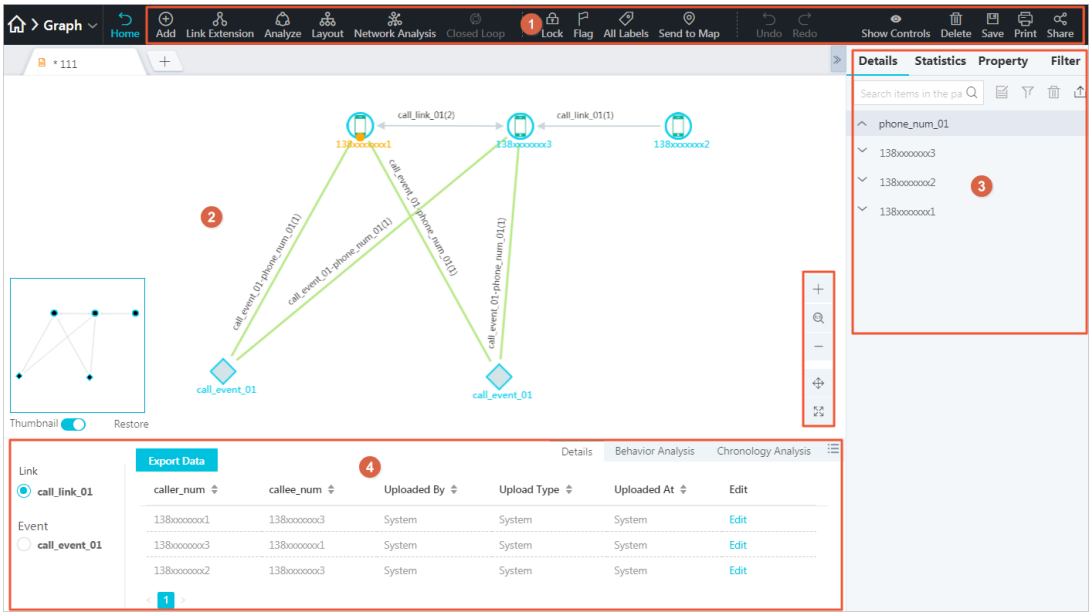
Graph is the operation interface of Analytics Workbench. You can create an analysis and view the analysis results in Graph. In this topic, you can learn about the features, function modules, and other information of Graph.

Graph is the core module of Analytics Workbench. This module covers most of the analysis and decision-making scenarios. The Graph module displays the relationship topologies among all objects. You can perform business computing and interactive graphic operations among multiple objects, and arrange visualized layouts based on your needs.

Meanwhile, user spaces and information cube networks are used as the assisting components to cover a wider range of analysis scenarios.

The **Graph** interface includes the following four areas.

Graph



- Area 1: functions
- Area 2: main graph area
- Area 3: properties and statistics
- Area 4: behavior analysis and chronology analysis

8.12.2. Analysis types

Graph Analytics supports four types of analyses: temporary analysis, common analysis, shared file analysis, and import data analysis.

Temporary analysis

An analysis that has just been created by the user is known as a temporary analysis. Temporary analyses have the same operations and features as other types of analyses, but temporary analyses cannot be shared before they are saved.

Common analysis

After a new analysis is saved and opened again, it is called a common analysis. The common analysis is the most commonly used analysis in Graph Analytics.

Shared file analysis

The analysis of shared files has the following statuses:

- Shared file - initial file

  After a common analysis is shared, it becomes a shared file. Each shared file will generate an initial file that is consistent with the original common analysis.

- Shared file - history analysis (draft)

  After the initial file has been edited by the shared member, a history analysis, also known as draft analysis, will be generated.

- Shared file - merged file

The system automatically merges multiple historical analyses into one analysis. Users can also merge the analyses manually.

Import data analysis

The import data analysis imports data in the **Data List** to the graph area to perform analyses. This type of analysis cannot be saved.

8.12.3. Create analyses

After you log on to Analytics Workbench, you must create an analysis and add the objects to be analyzed as nodes before you analyze the nodes.

Prerequisites

- Make sure that you have obtained an account and a password with the permission to perform graphic operations.
- Make sure that you have created source tables, objects, links, and events.
- Make sure that you have obtained data in the tables that have been mapped to the primary keys of the objects to be analyzed. You can obtain the data by querying the corresponding tables in the database.

Procedure

1. Log on to Analytics Workbench.
2. Click **Create Analysis**. A **Temporary Analysis** tab page appears.
3. Click **Add** in the toolbar and then click the blank space, or right-click the blank space and select **Add Node**. Set the parameters in the **Add Node** dialog box that appears.

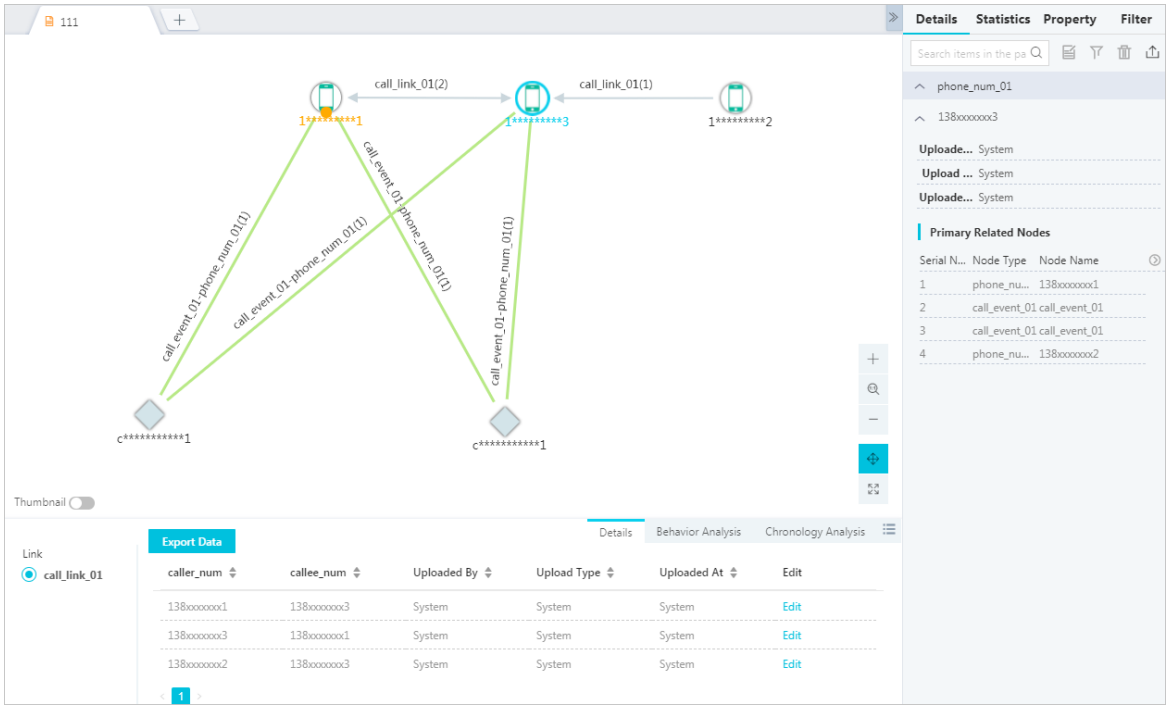
   The parameters are described in Parameters descriptions for adding a node.

Parameters descriptions for adding a node

Parameter	Description
Object type	The drop-down list displays all created objects. Select an object as needed. **Note** Graph Analytics supports adding compound nodes. A compound node is defined by multiple sub-types. For example, you can specify two sub-types for the object type "person": ID card and passport. The person can be uniquely identified by either the ID card or the passport.
Text area	Enter one or more primary key values. Separate multiple primary key values with commas (,).

4. Click **OK**.
5. Right-click a node that has been added, and select **Quick Extension**. The system automatically performs a link analysis based on the configured data sources, objects, links,

and events, and displays the analysis results in a graph.



6. Select one or more objects, links, or events. Click **Behavior Chronology** in the lower-right corner to see the corresponding **Details**, **Behavior Analysis**, and **Chronology Analysis** information.
7. Select one or more objects, links, or events. Click the icon () in the upper-right corner of the right-side pane to see the corresponding information on the **Details**, **Statistics**, **Property** and **Filter** tabs.
8. After the analysis has been completed, click **Save** in the upper-right corner. In the **Save Analysis** dialog box, enter a **File Name** and select a folder, and then click **OK**. A success message is displayed after the file has been saved.

   After you have saved the analysis file, if a collaborative analysis is required, you can share this personal analysis with other members.
9. Click the **Share** icon in the upper-right corner to specify the members you want to share this analysis with.

8.12.4. Add a node

Data analyses are based on nodes. Before you perform a data analysis, you need to add the object to be analyzed as a node.

Prerequisites

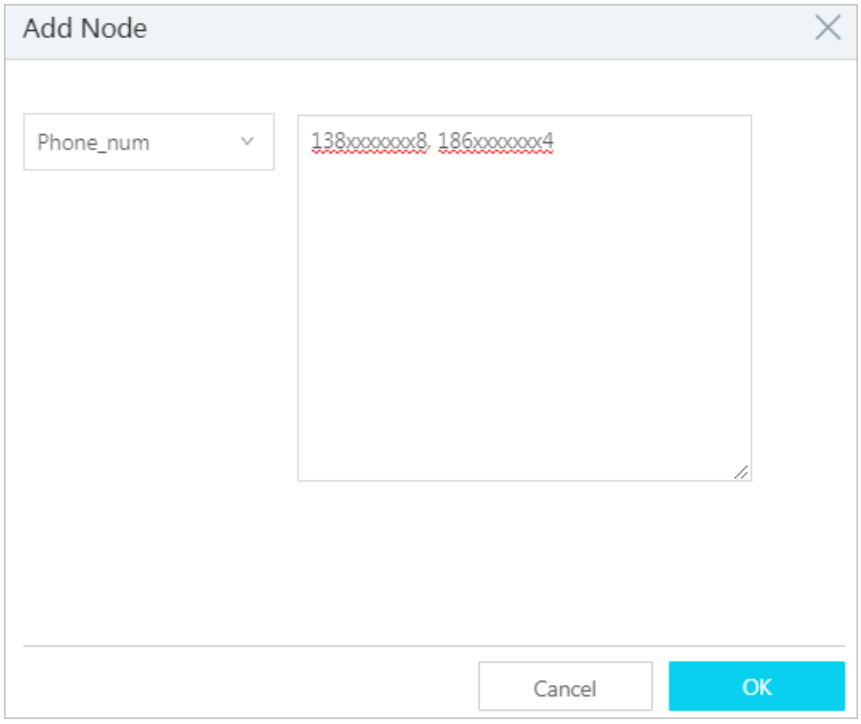
- Make sure that you have obtained an account and a password with the permission to perform graphic operations.
- You have created a source data table, object definitions, link definitions, and event definitions. For more information about these operations, see Data sources and Object information.
- Make sure that you have obtained data in the tables that have been mapped to the primary

keys of the objects to be analyzed. You can obtain the data by querying the corresponding tables in the database.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis file.
3. Click **Add Node** and click anywhere in the blank space. Or right-click anywhere on the blank space, select **Add Node**, and then set the parameters in the **Add Node** dialog box.

   The parameters are described in Parameter descriptions.



Parameter descriptions

Parameter	Description
Object Type	Displays all created objects. You can select an object as needed. Note Graph Analytics supports adding compound nodes. For example, you can specify two sub-types for object type "person": ID card and passport. The person can be uniquely identified by either the ID card or the passport.

Parameter	Description
Text Area	Enter one or more primary key values in the data table corresponding to the Object Information. Multiple primary key values must be separated by commas (,).

4. After you have configured the parameters, click **OK**.

   If you want to add only one node, you can click Add Node in the toolbar and click the graph area to place the node. Or, you can right-click anywhere in the blank space and select Add Node in the short-cut menu to place the node. When you add multiple nodes, the nodes are placed in a horizontal line layout or a matrix layout by default. You can configure the layout in **Advanced** > **System Settings** > **Functional Components** > **Relationship Network** > **add multiple node layouts**.

5. Click **Save** in the upper-right corner. In the **Save Analysis** dialog box that appears, enter the **File Name** and select a file directory. Click **OK**. A message is displayed, indicating that you have saved the file.

8.12.5. Delete nodes, links, and events

You can delete the selected content in the current graph area. You can quickly delete the nodes, links, and events.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Context

When you delete object nodes, links and events related to these object nodes are also deleted.

Procedure

1. Log on to Analytics Workbench, and open an existing analysis file, or create a new analysis.
2. You can delete object nodes, links, or events by using the following methods.

Method	Operation
In the toolbar	Select one or more nodes, links, or events in the canvas. Click the **Delete** icon in the toolbar. A message is displayed, indicating that the item has been deleted.
In the right-click menu	Select one or more object nodes, links, or events in the canvas, right-click a selected item, and select **Delete Selected**. A message is displayed, indicating that the item has been deleted.

8.12.6. Link extension

You can use link extension to search for all objects that are related to a specific object. You can discover associated clues and intelligence from large amounts of unrelated information, and turn the information into useful intelligence.

Prerequisites

- An analysis file exists. Add a new analysis as shown in Create analyses.
- A node object exists. Add a new node as shown in Add a node.

Context

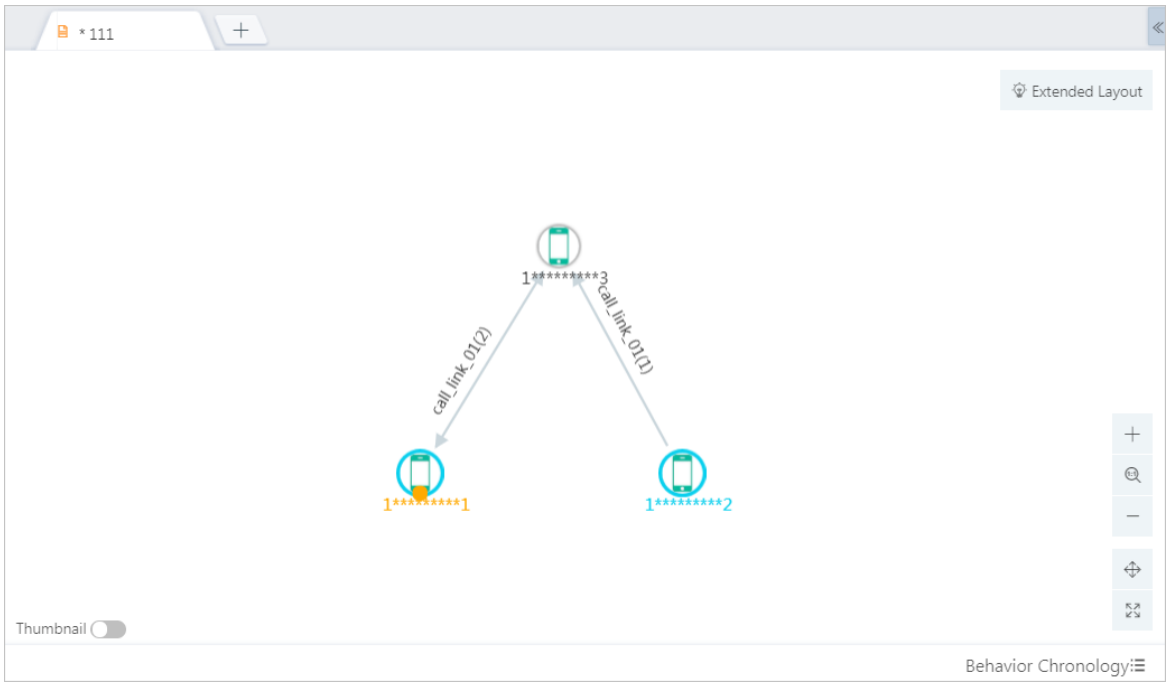
Link extension supports the following two modes.

Link extension mode	Description	Reference procedure
Simple mode	Double-click a node to perform a first-degree link analysis on the node. Only the associated information for the first-degree link can be extended.	Step 3
Advanced mode	Filter specific conditions based on business experience to extend the associated information.	Step 4 to Step 7

Procedure

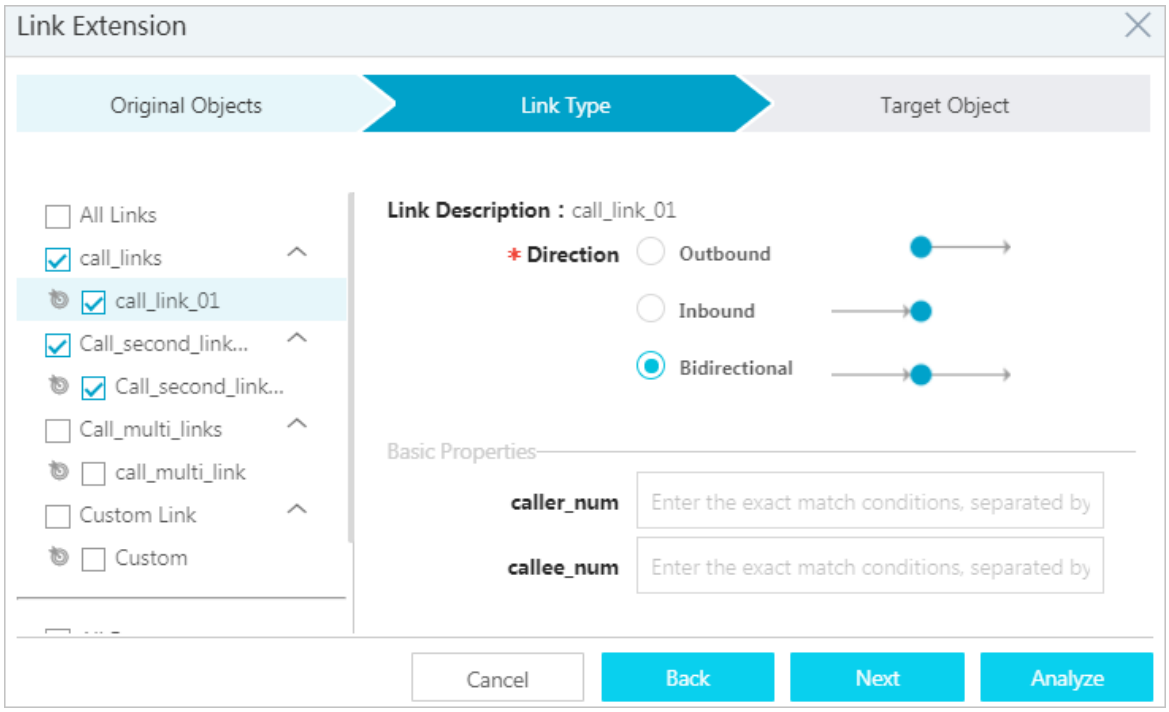
1. Log on to Analytics Workbench.
2. Click an existing analysis file to open the file on the Graph page, or click **Create Analysis** to create an analysis file and add nodes.

   Perform a simple analysis on a specific node.
3. Double-click a node to perform a first-degree link analysis.

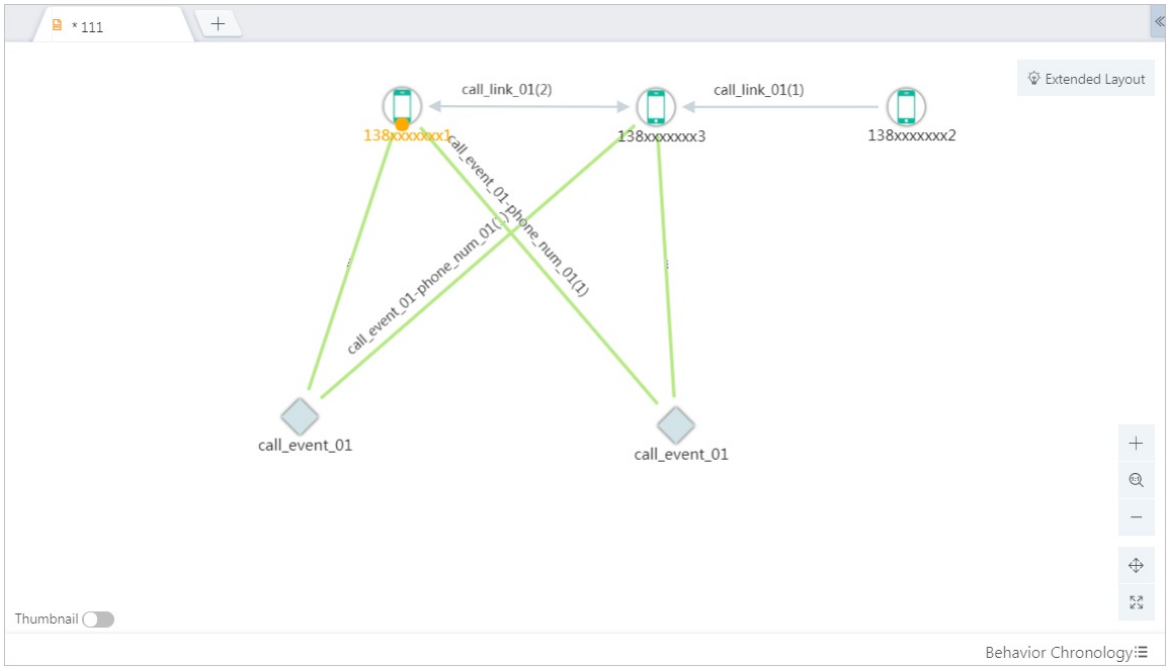


   Perform an advanced analysis on one or more nodes.

4. Select one or more nodes and click the **Link Extension** icon in the toolbar. You can select a maximum of 1,000 nodes. In the **Link Extension** dialog box, select the source objects.
5. Click **Next**, select links and the events, and specify the parameters for the links and events.



6. Click **Next**, and select the target objects.
7. Click **Analyze** to extend the link of the selected nodes.



8.12.7. Graphic operations

8.12.7.1. Move canvases

You can move the entire graph area by moving the canvas to perform comparative analysis.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Procedure

1. Log on to Analytics Workbench, and open an existing analysis file or create a new analysis.
2. You can move the canvas by using the following methods.

> **Note** After you move the canvas, it will take a short time to reproduce the canvas. There will be a short delay. During that time, do not operate the canvas so as to prevent unexpected results.

Method	Operation
In the tool bar	Click the **Enter Move Canvas Mode** icon in the lower-right corner of the graph area and drag the canvas with your mouse.
Space key + mouse	Long press the **Space** key and drag the canvas with your mouse.
Use the scroll wheel	Move the canvas upwards or downwards with the scroll wheel of your mouse.

8.12.7.2. Zoom in and zoom out canvases

Graph Analytics enables you to zoom in and zoom out the content in the graph area to perform a comparative analysis from a full-graph or partial-graph perspective.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Procedure

1. Log on to Analytics Workbench, and open an existing analysis file or create an analysis.
2. You can zoom in or zoom out the canvas in the following methods.

> **Note** After you zoom in or zoom out the canvas, it will take a short time to reproduce the canvas. There will be a short delay. During that time, do not operate the canvas to prevent unexpected results.

Method	Operation
Shift key + mouse pointer	Long press the **Shift** key and zoom in or zoom out the canvas with the scroll wheel of your mouse.

Method	Operation
Interface icons	○ Click the **Zoom In** icon to scale the size of the canvas. ○ Click the **Zoom Out** icon to scale the size of the canvas. ○ Click the **Restore Original Ratio** icon to restore the original ratio of the canvas.

8.12.7.3. Undo and redo operations

Graph Analytics supports undo and redo operations. In case of a mistake, you can quickly restore the graph area to the status prior to the operation.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Background information

The undo and redo functions are described as follows:

- Undo: You can temporarily store the graph area status prior to the operation into the operation history. You can restore a historical version by undoing the operation. A maximum of 20 steps can be undone.
- Redo: After the Undo operation, you can also restore the analysis to a status prior to the undo operation.

The undo and redo operations can be performed to add nodes, add links, save virtual nodes, merge or split nodes. Undo and redo operations are applicable to layout features, link analysis features, and paste operations, and can be used to delete nodes or links.

Procedure

1. Log on to Analytics Workbench, and open an existing analysis file or create a new analysis to perform multiple operations.
2. If you need to go back to a specific status, you can undo the operation by using the following methods.

Method	Description
Undo	In the toolbar, click the **Undo** icon to undo the operation and restore the area to the status prior to the operation. You can perform the undo operation at a maximum of 20 steps.

Method	Description
Redo	In the tool bar, click the **Redo** icon to restore the graph area to the status prior to the undo operation. You can perform the redo operation at a maximum of 20 steps.

8.12.7.4. View thumbnails

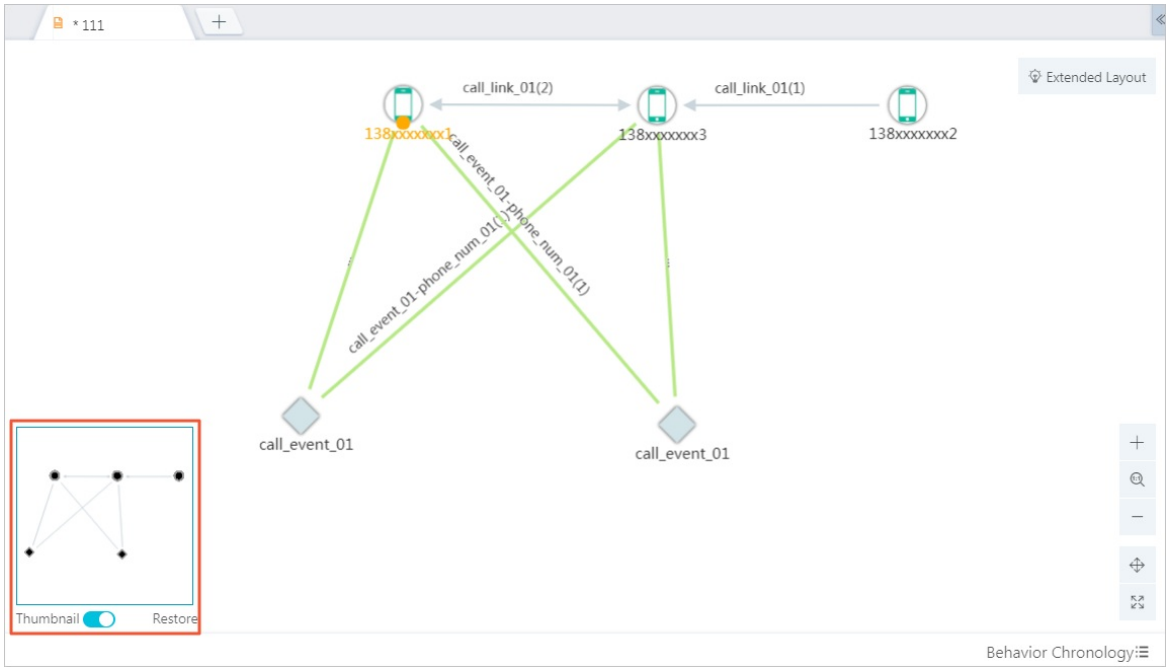
The thumbnail tab is located in the lower-left corner of the graph area. After you click the thumbnail tab, the thumbnail of the current full graph is displayed. The position of the visible area is framed out to help you view the analysis graph easily.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis.
3. Click the **Thumbnail** icon in the lower-left corner to open the thumbnail of the current graph area.



4. You can drag the rectangular box in the middle of the thumbnail to drag the corresponding nodes in the canvas.

8.12.7.5. Right-click menu

The right-click menu in the graph area enables you to quickly operate on nodes, links, or events. In this topic, a vertex refers to a node, and an edge refers to a link or an event.

Log on to Analytics Workbench, and open an existing analysis file or create an analysis. The right-click menu varies depending on the position where you click on.

Right-click anywhere on the blank space

Right-click anywhere on the blank space. For more information, see Right-click menu description.

Right-click menu description

Menu option	Description
Add Node	Adds nodes to the graph area. For more information, see Add a node.
Select All Nodes	Selects all nodes in the graph area.
Select All Events	Selects all events in the graph area.
Select All Links	Selects all links in the graph area.
Paste	Pastes the content in the clipboard to the current analysis. This option will be grayed out if no objects or links exist in the clipboard.

Right-click a node or a link

Right-click a vertex or an edge in the graph area. For more information, see Right-click menu description.

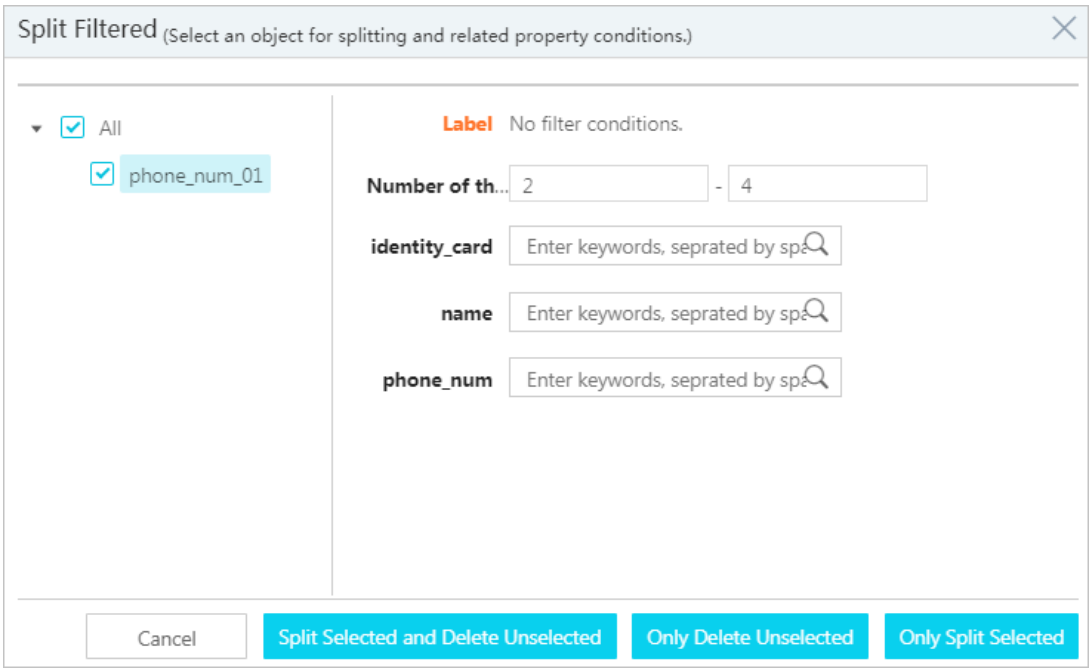
Right-click menu description

Menu option	Description
Add Link	Adds a link for the selected two nodes to establish a relationship network. Graph Analytics supports directed and undirected links. • Directed link: A clearly defined link with specific direction. The link line usually has arrows, as in the case of phone call relationship. • Undirected link: A link with no clear direction. For example: people taking a train.
Add Label	Adds label information to the selected nodes for easy analysis and identification. For more information about labels, see Add user labels.
Delete Selected	Deletes the selected nodes or links from the graph area. For more information, see Delete nodes, links, and events.
Inverse Selection	Selects all nodes other than the currently selected nodes.
Select Correlated Nodes	Selects all nodes that are correlated to the currently selected nodes. This option helps users locate the information for analysis and quickly select the correlated nodes. This option applies only to object nodes.

Menu option	Description
Merge Selected	Merges two or more selected nodes and displays the results. Users can sort, merge, visualize, and simplify the information in the entire graph area. This option applies only to object nodes, and the number of selected nodes must be greater than or equal to 2. Meanwhile, after the Merge Selected option is applied, the object nodes and links in the Selected state in the graph area remain unchanged.
Split Selected	Splits the merged nodes and displays the results. Users can extract information from a merged node for individual analysis. This option applies only to object nodes. Make sure that at least one node in the selected nodes is part of a merged node. Meanwhile, after the Split Selected option is applied, the object nodes and links in the Selected state in the graph area remain unchanged. The split operation filters the label information and property information of merged nodes by type. An UI example is shown in Split Filtered dialog box, and the parameters are described in Filter descriptions. The split options are described in Split options.
Event Chain	Displays all events that occur on a specific object within a specified time range.
Clear Event Chain	Clears all event chains of an object within a specified time range.
Show Event	Shows an event. When an event involves multiple objects, you can use this operation to show the event and view all objects involved in the event.
Hide Event	Hides an event that has been shown.
Quick Extension	Performs a quick extension on the selected nodes, obtains the graphic results of the quick extension, and makes force-directed layouts.
Node Redirect	Redirects to a third-party business system by node type according to the URL configured in Administration Console. **Node Redirect** is a channel that connects Graph Analytics to other products. For example, if the object information can be viewed only by a third-party system, you can click **Node Redirect** to view the information in the third-party system. Configure the redirect URL in Administration Console depending on the specific situation.
Copy	Copies the selected object nodes or link graphs and pastes them to the clipboard. You can copy object nodes and link graphs.

Menu option	Description
Roll Up	If a multi-degree link is established between two objects, multiple intermediary objects are displayed between the two nodes. Select these nodes and click **Roll Up** to hide these intermediary nodes. After the intermediary objects are hidden, a dotted line is displayed between the two objects to represent the multi-degree link. To show these intermediary objects, press **Ctrl** and click on the dotted line.
Save Virtual Node	Persists virtual nodes.
View Selected Node Details	You can view the details of a selected node in the right-side pane.

Split Filtered dialog box

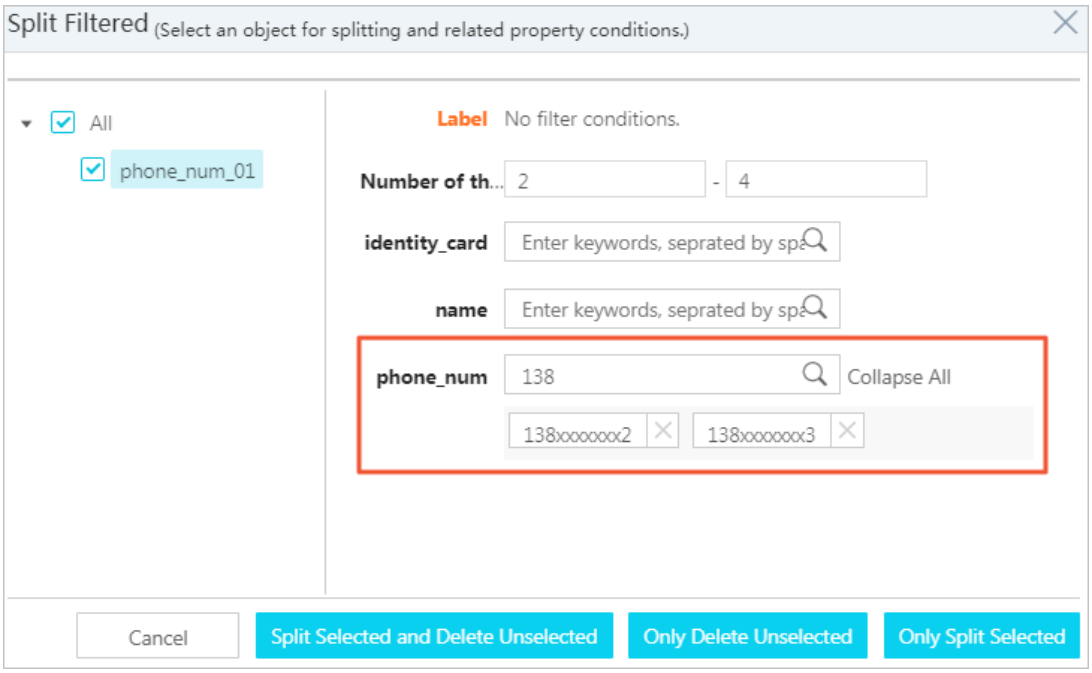


Filter descriptions

Filter	Description
Labels	All labels are enumerated. You can delete some labels as needed.
Time type	Lists the maximum and minimum values of the time property. You can adjust the value range, for example, the departure time.
Value type	Lists the maximum and minimum numerical values. You can adjust the value range, for example, age.
Dictionary type	Like the Label filter, all dictionaries are enumerated. You can delete some dictionaries as needed.

Filter	Description
String	Supports fuzzy searches. As shown in Fuzzy search, search for the mobile phone number containing 189. Each item in the search results can be deleted by clicking the Delete icon next to the item. If no results have been found, the message "No results have been found." is displayed.

Fuzzy search



Split options

Option	Description
Cancel	Cancels the split operation.
Split Selected and Delete Unselected	Splits the nodes that meet the conditions, and deletes the other nodes.
Only Delete Unselected	Retains nodes that meet the conditions in the merged node and deletes the other nodes.
Only Split Selected	Splits nodes that meet the conditions from the merged node, and retains the other nodes in the merged node.

8.12.8. Analyze

8.12.8.1. Group Analysis

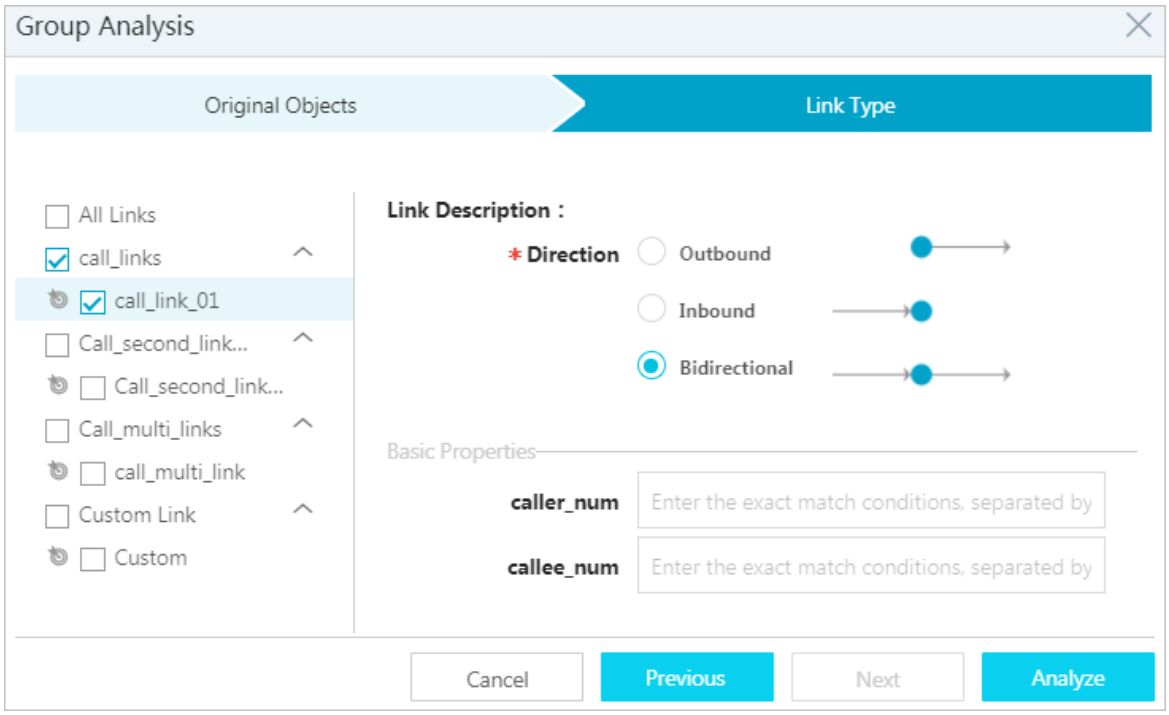
You can use group analysis to perform multiple analyses on the relationships between any two objects in the group.

Prerequisites

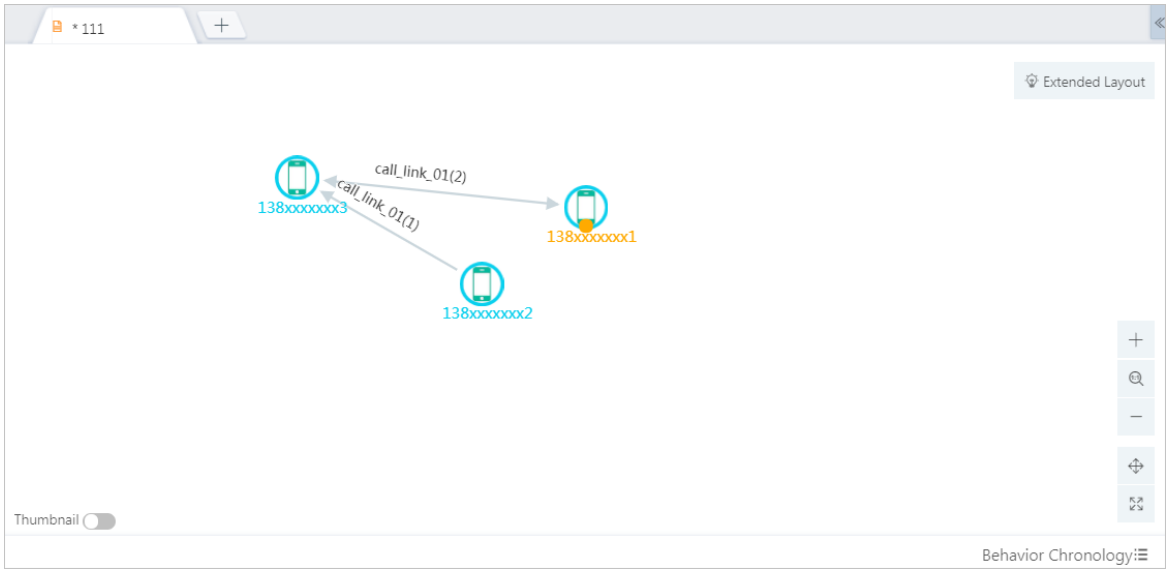
- Make sure that you have obtained an account and a password with the permission to perform graphic operations.
- An analysis file exists. Add a new analysis as shown in Create analyses.
- Two or more node objects already exist. Add a new object node as shown in Add a node.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select two or more object nodes in the graph area.
3. Choose **Analyze** > **Group Analysis**. In the **Group Analysis** dialog box that appears, select the start object to be analyzed.
4. Click **Next**, select links, and set conditions for each link based on your needs.



5. Click **Analyze** to complete a group analysis for the selected node.



8.12.8.2. Common neighbor analysis

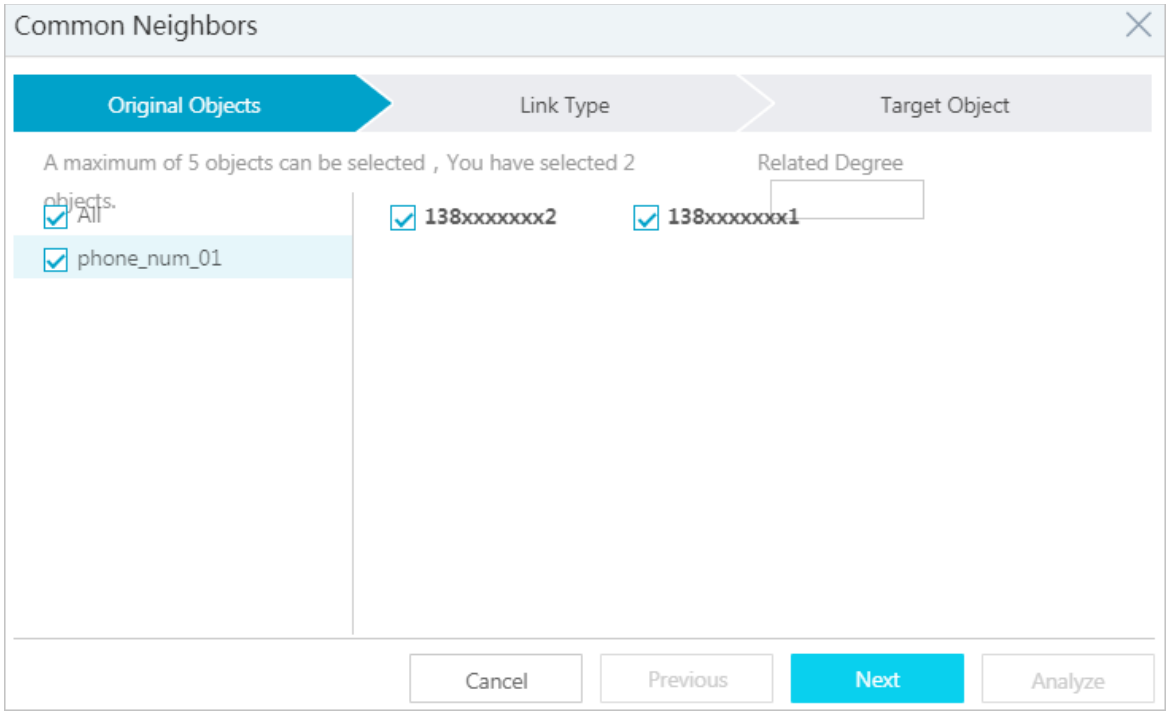
This feature allows you to analyze the commonly associated objects for a group of identical objects or different objects. The correlated degree is set to 2 by default.

Prerequisites

- You have obtained an account and a password with the permission to perform graphic operations.
- An analysis file exists. For more information about how to add an analysis, see Create analyses.
- Two or more node objects already exist. For more information about how to add a node, see Add a node.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select two object nodes in the graph area.
3. In the toolbar, choose **Analyze** > **Common Neighbors**. In the **Common Neighbors** dialog box that appears, select the start object to be analyzed.

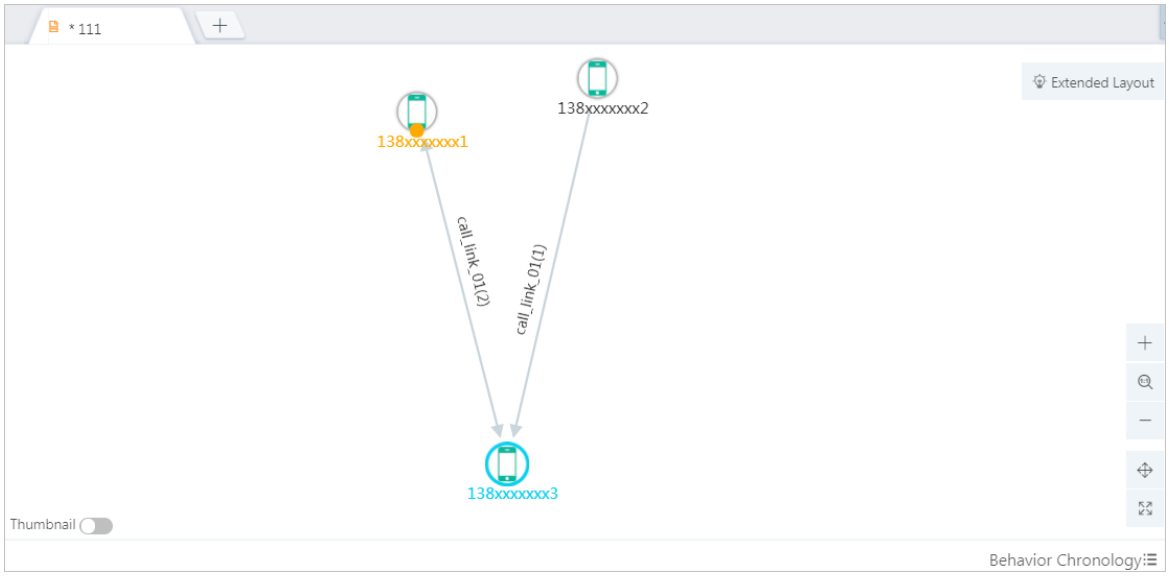


4. Click **Next** to select links and set conditions for each link based on your needs.

   The correlated degree is set to 2 by default.

   You can set the link conditions by time, date, value, enumerated type, string equality, and fuzzy string matching.

5. Click **Analyze** to complete a common neighbor analysis for the target node.



8.12.8.3. Lineage analysis

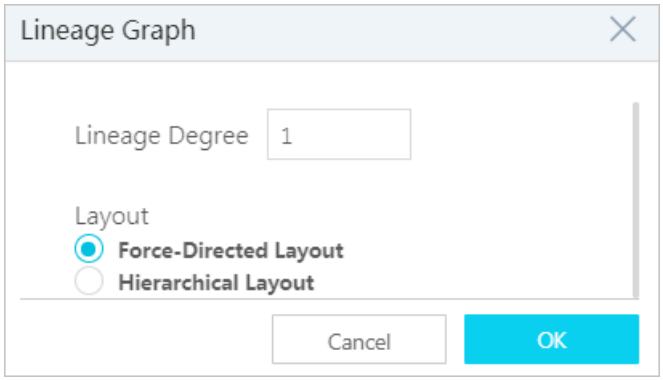
Lineage analysis allows you to search the lineage between certain types of objects based on the lineage configurations in Administration Console.

Prerequisites

- You have obtained an account and a password with the permission to perform graphic operations.
- An analysis file exists. For more information about how to add an analysis, see Create analyses.
- A node object exists. For more information about how to add a node, see Add a node.

Procedure

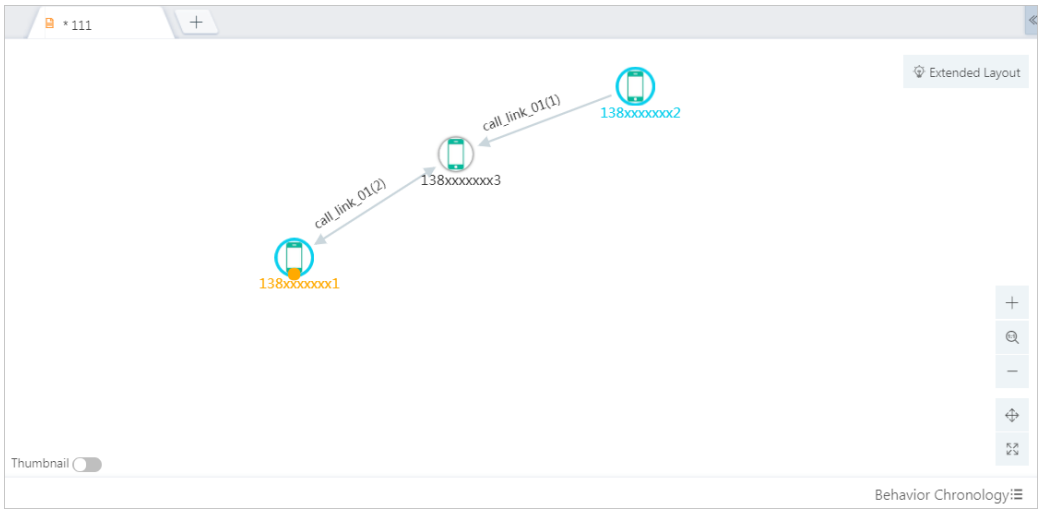
1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select one or more nodes in the graph area to perform a lineage analysis.
3. In the toolbar, choose **Analyze** > **Lineage Analysis**. In the dialog box that appears, specify Lineage Degree and Layout.



4. Click **OK** to complete the lineage analysis of the target node.

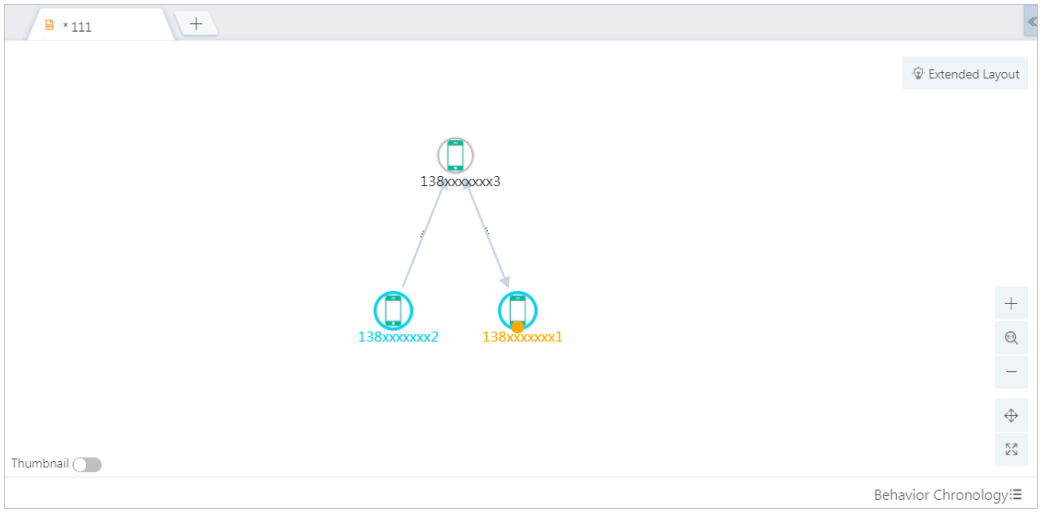
   The Lineage Degree is *1*. The Layout is *Force-Directed Layout*. The result is shown in Analysis results in the force-directed layout.

   Analysis results in the force-directed layout



   Assume that you have set the first-degree link and selected the hierarchical layout. The analysis results are shown in Analysis results in the hierarchical layout.

   Analysis results in the hierarchical layout



8.12.8.4. Path analysis

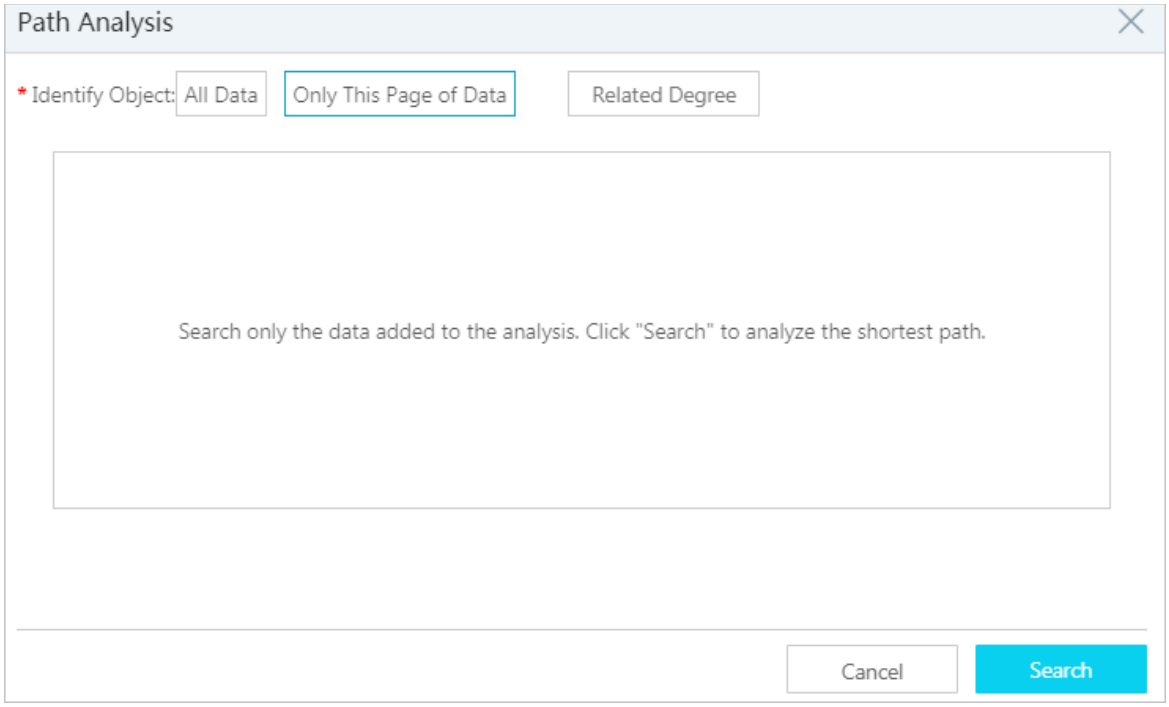
This feature allows you to analyze the link path between two objects.

Prerequisites

- You have obtained an account and a password with the permission to perform graphic operations.
- An analysis file exists. For more information about how to add an analysis, see Create analyses.
- Two or more node objects already exist. For more information about how to add a node, see Add a node.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select two object nodes in the graph area.
3. In the toolbar, choose **Analyze** > **Path Analysis**. The **Path Analysis** dialog box appears.



The path analysis supports performing analyses on All Data and Only Data of This Page. For more information about the path analysis, see the following procedure.

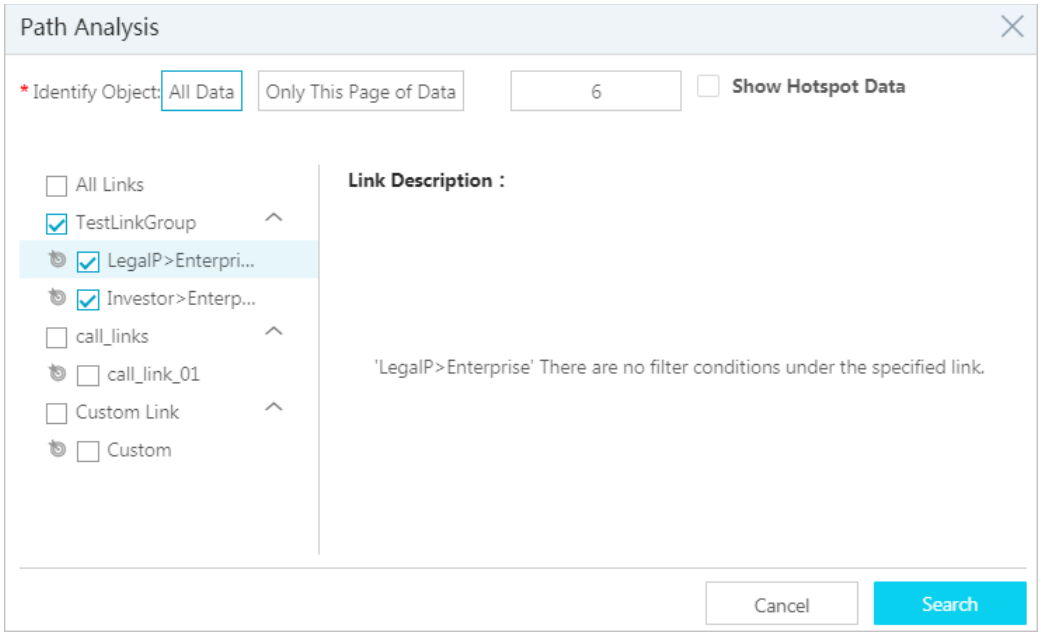
4. Perform path analysis on all data: Select **All Data** to perform a path analysis and specify the relevant parameters.

   All Data: Set the link condition, correlated degree, and whether to show the hotspot data, as shown in Set conditions for the path analysis on all data.

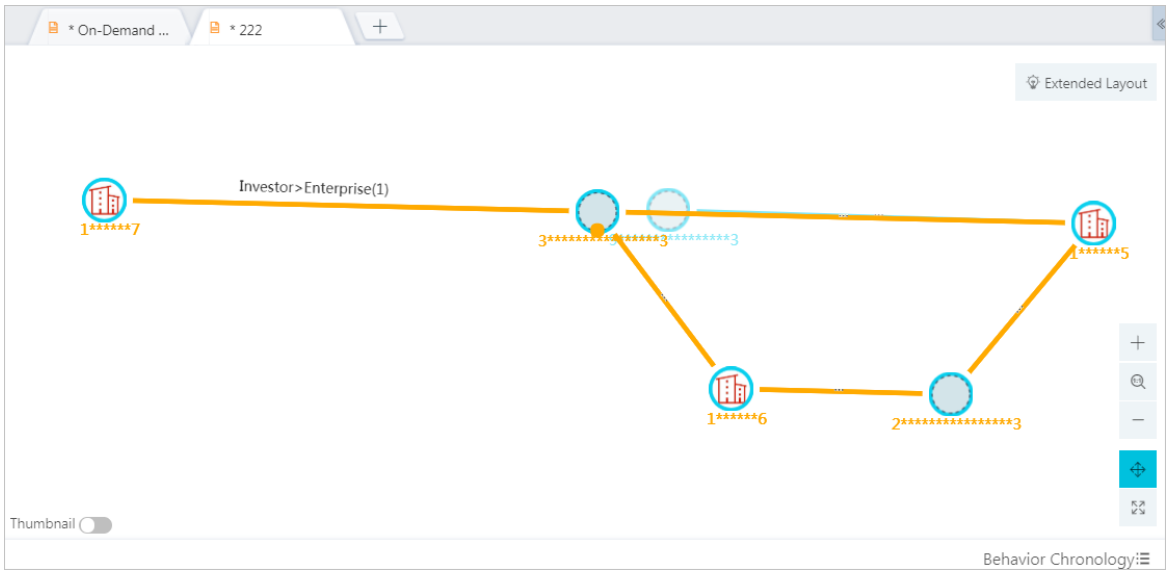
   If you select Show Hotspot Data, the hotspot data is displayed but not extended.

   > **Note** The system administrator monitors the hotspot data and adds a red tab to the hotspot node to inform the users of the hotspot data.

   Set conditions for the path analysis on all data



5. Click **Search** to perform a path analysis on all data.



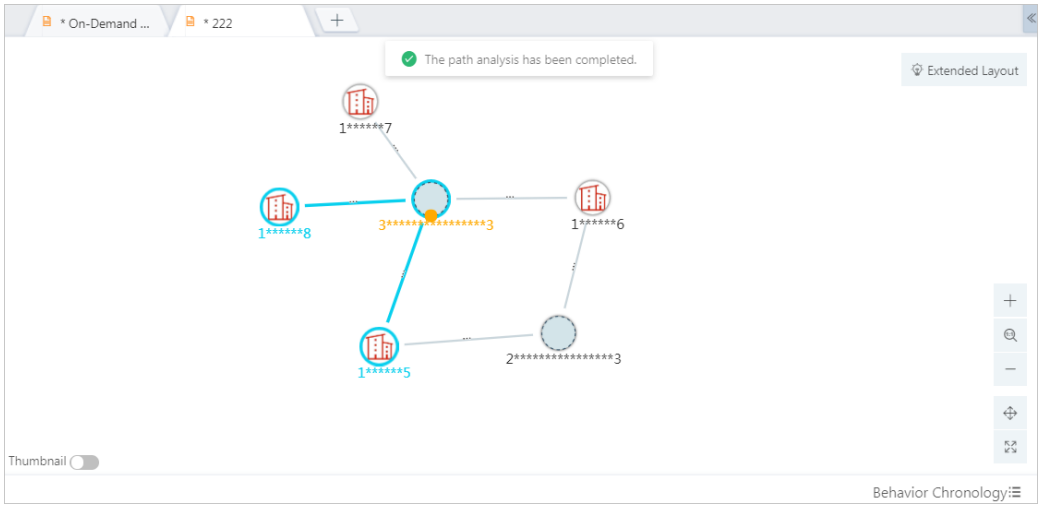
6. Analyze only data of this page: Select **Only Data of This Page** and specify the correlated degree.

   Analyzing only data of this page is to analyze the path between two nodes on the current page. This feature supports analyzing the shortest path and paths with a specified correlated degree.

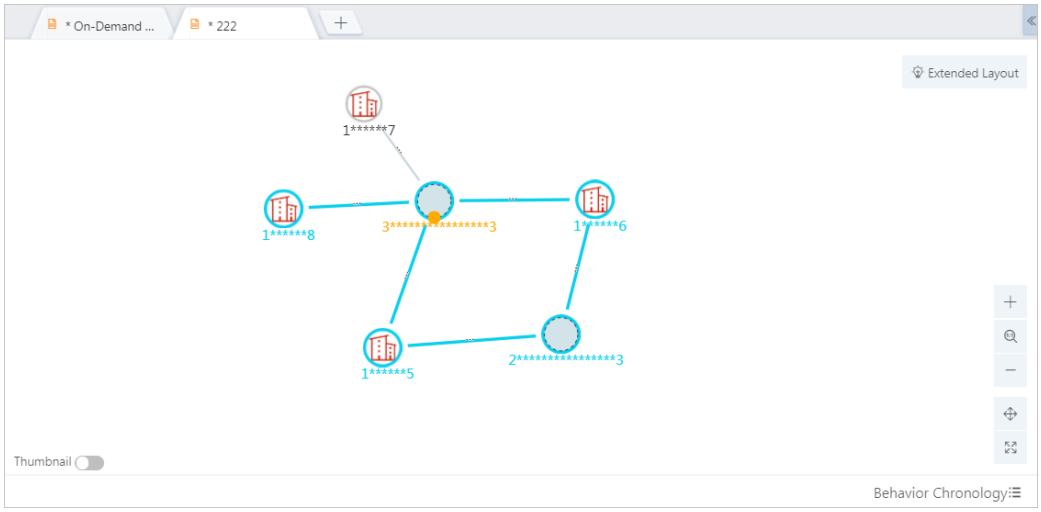
   Correlated Degree: If this parameter is not specified, the shortest path is analyzed by default. If you enter N, all paths with degrees less than or equal to N will be searched. N is specified by Administration Console and cannot be higher than 6.

7. Click **Search** to perform a path analysis on the data of this page.

   Results of shortest path analysis - only data of this page



Results of the path analysis with a specified correlated degree - only data of this page



8.12.8.5. Backbone analysis

Based on the membership network on the current page, the backbone analysis uses smart service algorithms to help you explore the key nodes in a relationship network.

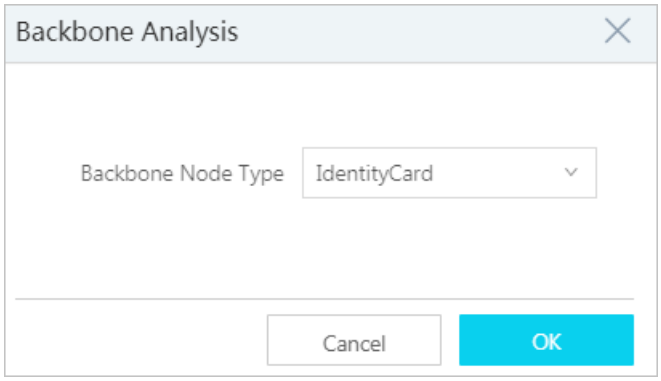
Prerequisites

- You have obtained an account and a password with the permission to perform graphic operations.
- An analysis file exists. For more information about how to add an analysis, see Create analyses.
- An object node already exists. For more information about how to add a node, see Add a node.

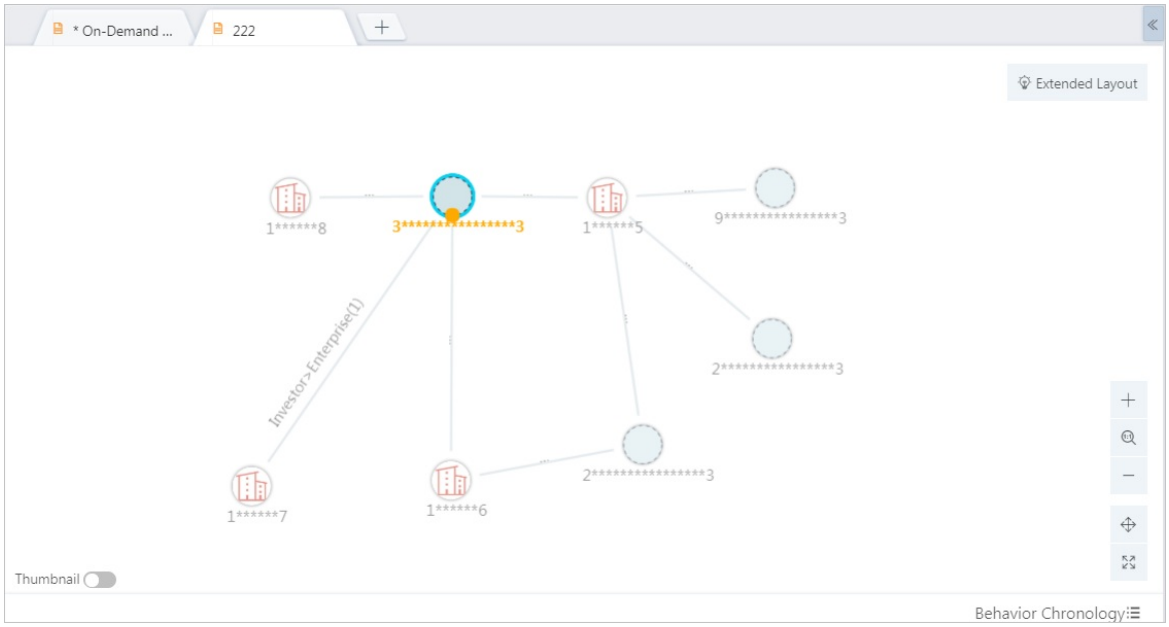
Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select an object node in the graph area.

3. In the toolbar, choose **Analyze** > **Backbone Analysis**, and set the Backbone Node Type in the dialog box that appears.



4. Click **OK**, and the key nodes in the graph area are highlighted.



8.12.8.6. Intimacy measurements

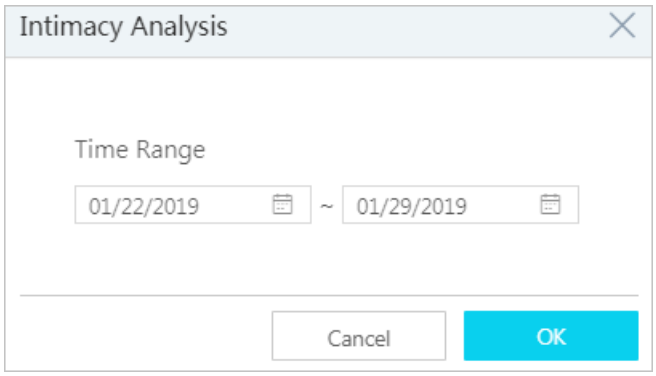
Perform an intimacy measurement to query the intimacy among objects of a specific type based on the intimacy settings configured in Administration Console.

Prerequisites

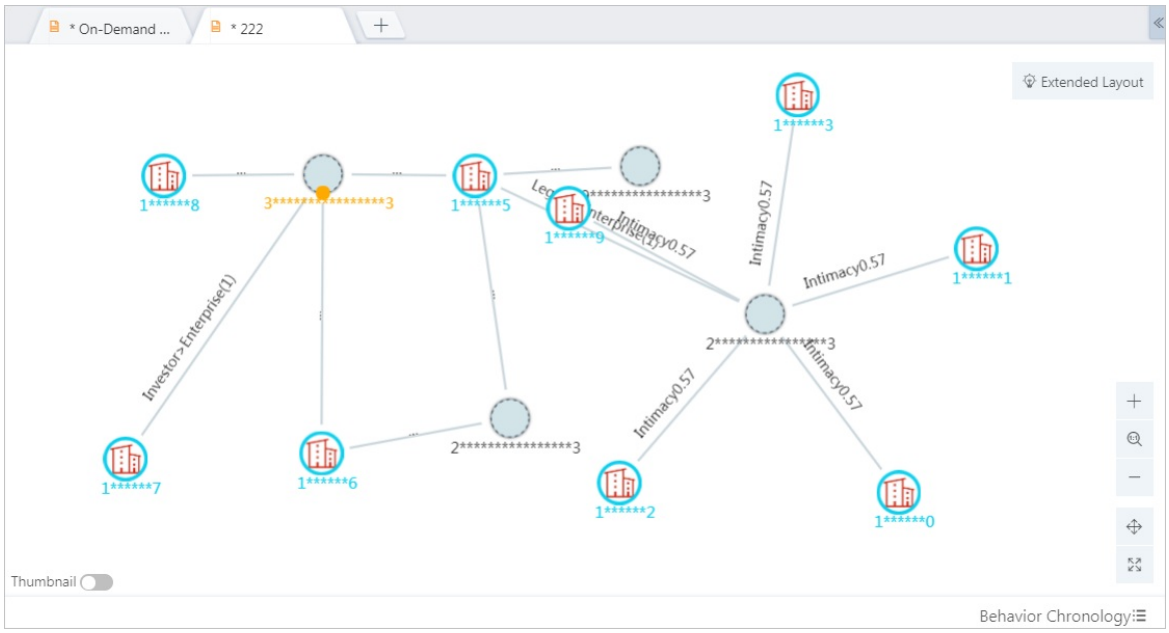
- Make sure that you have obtained an account and a password with the permissions to perform graphic operations and intimacy measurements.
- Make sure that you have configured intimacy settings in Administration Console. For more information, see Intimacy measurement settings.
- An analysis file already exists. For more information about how to create a new analysis, see Create analyses.
- An object node already exists. For more information about how to add a new node, see Add a node.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create an analysis, and select one or more nodes of the same type that have been created in the graph area.
3. Choose **Analyze** > **Intimacy Measurement** from the toolbar. In the **Intimacy Measurement** dialog box, set the time range.



4. Click **OK** to perform an intimacy measurement on the selected nodes.



8.12.9. Lock or unlock nodes

The node locking function keeps the node in a fixed position in the canvas to facilitate your operations.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Context

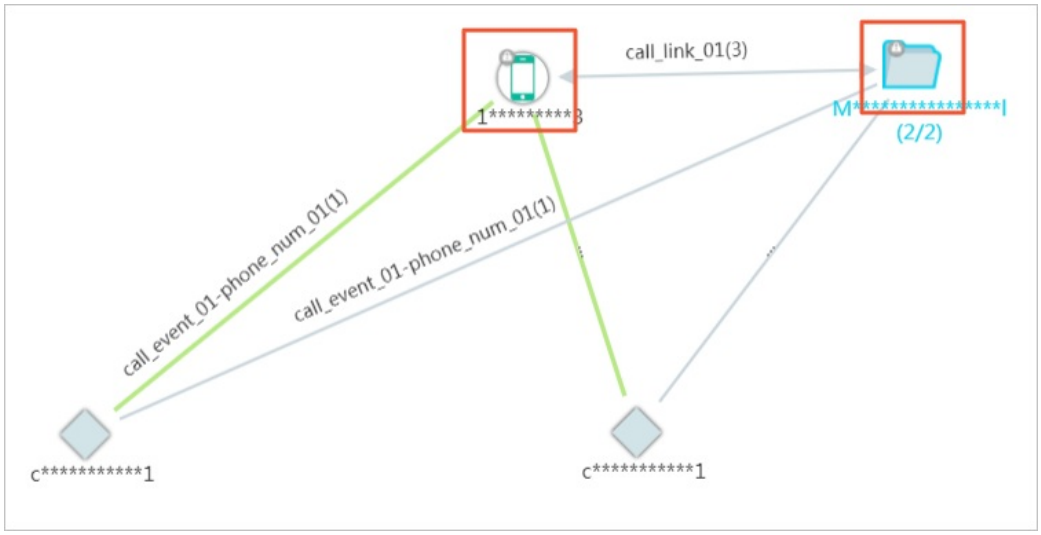
Except for nodes in the **Force-Directed Layout**, nodes in other layouts cannot be dragged. However, when the canvas moves as a whole, the locked nodes will move with the canvas.

Procedure

1. Log on to Analytics Workbench, and open an existing analysis file or create an analysis.
2. You can lock or unlock nodes by using the following methods.

Method	Operation
Lock nodes	In the canvas, select one or more nodes, including merged nodes, and click the **Lock** icon in the toolbar to lock the selected nodes or merged nodes. When a node is locked, you can see a gray lock at the top left of the node, as shown in Lock nodes.
Unlock nodes	In the canvas, select one or more nodes, including merged nodes, and click the **Unlock** icon in the toolbar to unlock the selected nodes or merged nodes. When a node is unlocked, the gray lock at the top left of the node disappears. **Note** If you split a locked merged node, the split nodes are automatically unlocked.

Lock nodes



8.12.10. Network analysis

Network Analysis can be used to analyze node relationships from multiple perspectives, such as location precedence, closeness, and activity frequency.

Prerequisites

You have obtained an account and a password with the permission to perform network analyses.

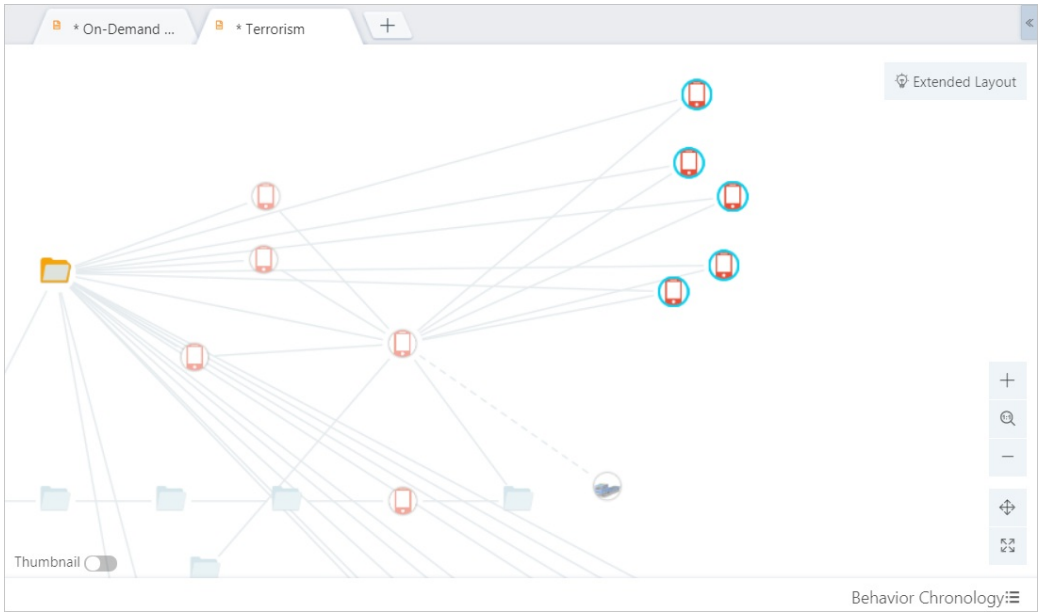
Context

The features of network analysis are described as follows:

- Closeness: Searches the key nodes that are most closely related to other parts of the network and checks the statuses of these nodes.
- Location precedence: Searches the bridging nodes that control the information flow.
- Activity frequency: Searches objects that are more active and more frequently interacted with other objects.
- Data flow direction: Searches the flow of relational data between objects in the network. It is only valid for directed relationships.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis, and select one or more object nodes in the graph area.

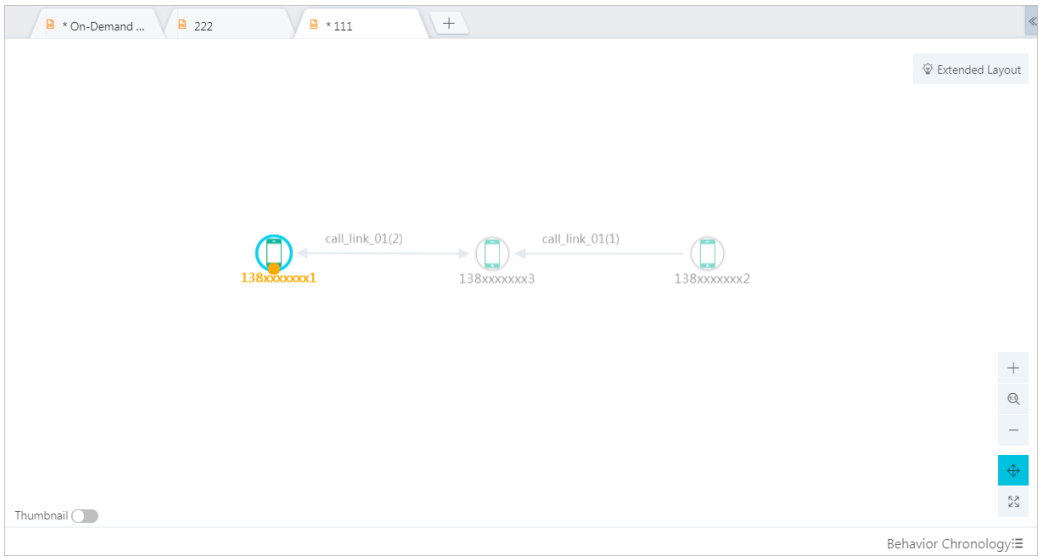


3. Example of location precedence analysis: In the toolbar, choose **Network Analysis** > **Location Precedence**, as shown in Location precedence analysis.

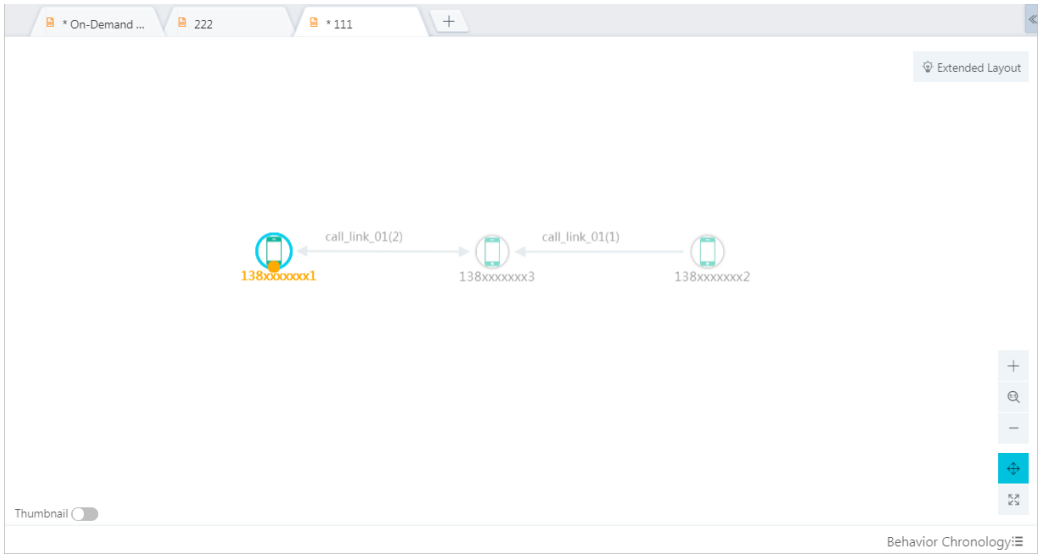
   Location precedence analysis

Other analysis examples are shown in Closeness analysis, Activity frequency analysis, and Data flow direction analysis.

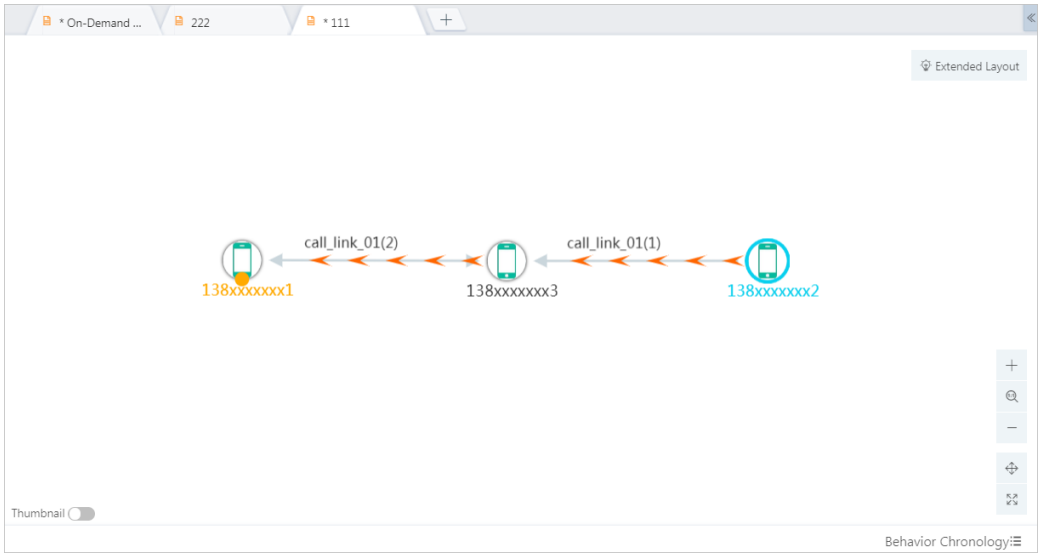
Closeness analysis



Activity frequency analysis



Data flow direction analysis



8.12.11. Closed-loop mining

Closed-loop mining allows you to check one or more nodes for continuous closed-loop links. This feature can be used to analyze cases including cash advance cases.

Prerequisites

- You have obtained the account and password with the permission to perform graphic operations.
- An analysis file already exists. For more information about how to create a analysis, see Create analyses.
- One or more node objects already exist. For more information about how to create a node, see Add a node.

Context

A closed loop refers to a link that starts from a node and ends at the same node. The following closed loops are available: the third-degree closed loops, fourth-degree closed loops, fifth-degree closed loops, and the sixth-degree closed loops.

The following example shows that a credit card cash advance will eventually return to the cashier regardless of the direction of the fund flow. For example, A is the cashier, and both B and C are coordinators. The fund flow is typically A > B > C > A, which forms a closed loop.

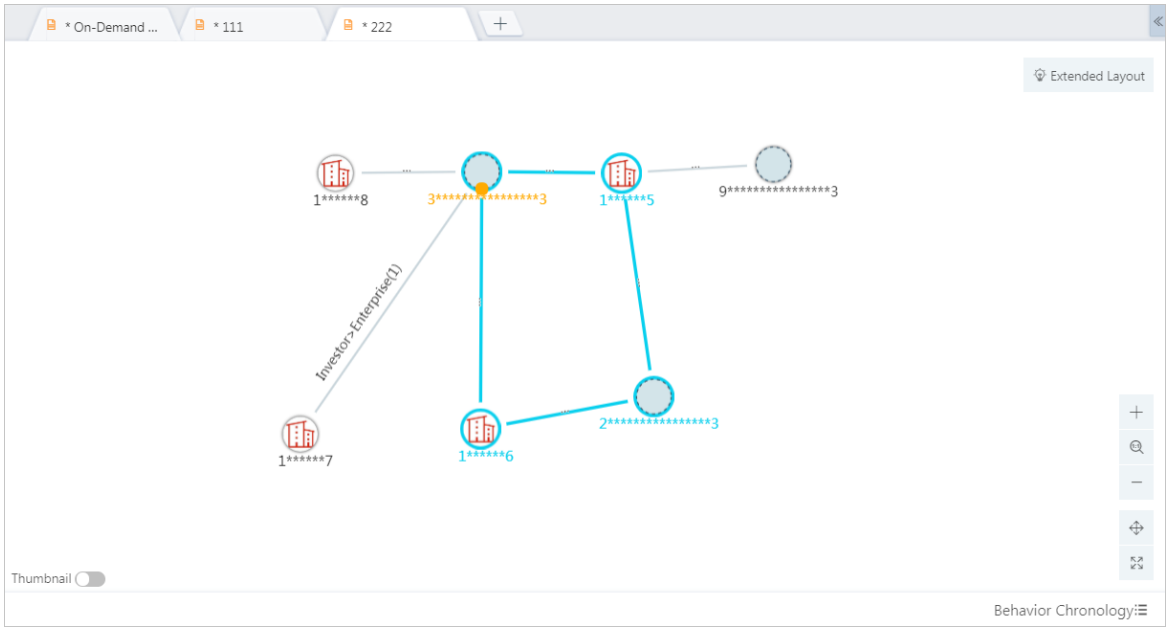
This topic describes loop steps by analyzing the third-degree closed loop of phone calls.

Procedure

1. Log on to Analytics Workbench.
2. Create an analysis and perform related analyses, or open an analysis file that already has analysis results.
3. Select a node in the analysis result, and choose **Closed-Loop > 3-Degree Closed Loop** in the toolbar.
4. In the dialog box that appears, specify **Directionality** and the link, and then click **OK**.

   Directionality has the following options:

   - *Undirected*: Analyzes both directional and undirectional closed-loop links on the specified node.
   - *Directed*: Analyzes only the directional closed-loop links on the specified node.



8.12.12. Layouts

In Graph Analytics, you can easily analyze the content in multiple layouts.

Prerequisites

You have obtained an account and password with **Layout** permissions.

Background

Supported layouts are as follows.

Layout	Description
Matrix Layout	Objects are arranged in a matrix structure to help you sort and organize information during the analysis process.
Circle Layout	Objects are organized and evenly arranged in a circle to display their relationships. This layout helps you sort information and analyze the key nodes during the analysis process.
Horizontal Layout	Objects are arranged along a horizontal line to help you analyze the information from a horizontal perspective.
Vertical Layout	Objects are arranged along a vertical line to help you analyze the information from a vertical perspective.
Force-Directed Layout	The force-directed layout is used to visualize complex networks. All edges are more or less of equal length and there are as few intersecting edges as possible. With nodes and the weights of edges defined in advance, the force-directed layout positions the nodes automatically according to the principle that a higher weight leads to shorter distance. This procedure is convenient for you to tell how close nodes are from each other. This is a global layout where all object nodes and links in the graph area of Graph Analytics are calculated.
Hierarchical Layout	Objects are arranged in a tree structure. The hierarchical layout is used for family trees and the organizational structures of enterprises.

Procedure

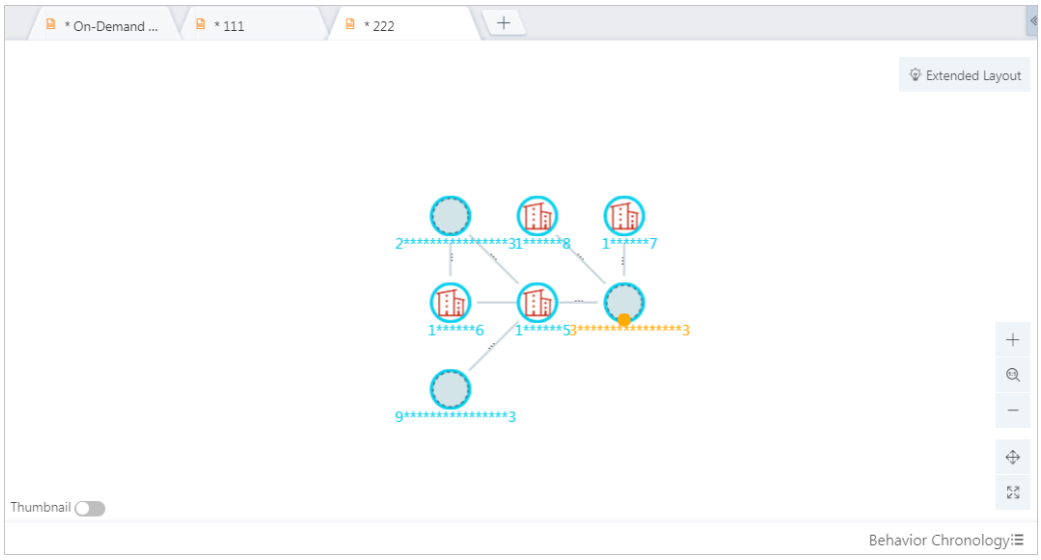
1. Log on to Analytics Workbench.
2. You can create a new analysis and produce the analysis results. Or, you can open an analysis file that already has the analysis results.
3. Take the matrix layout as an example. In the toolbar, choose **Layout** > **Matrix Layout**, as shown in Matrix layout.

   > **Note** We recommend that you choose a suitable layout for your business, so that you can easily view the analysis data.

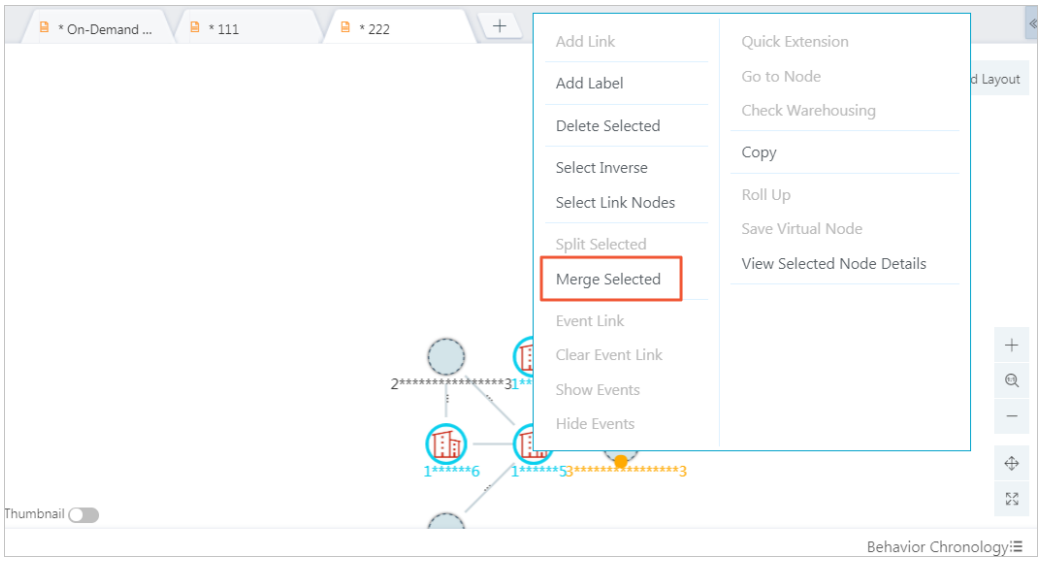
   The matrix layout can be used with the Merge Nodes feature, as shown in Merge nodes. This operation can sort and merge the analysis results when you hand large amounts of information. The merged result is shown in Merged result.

   Other layouts are shown in Circle layout, Horizontal layout, Vertical layout, Force-directed layout, and Hierarchical layout.

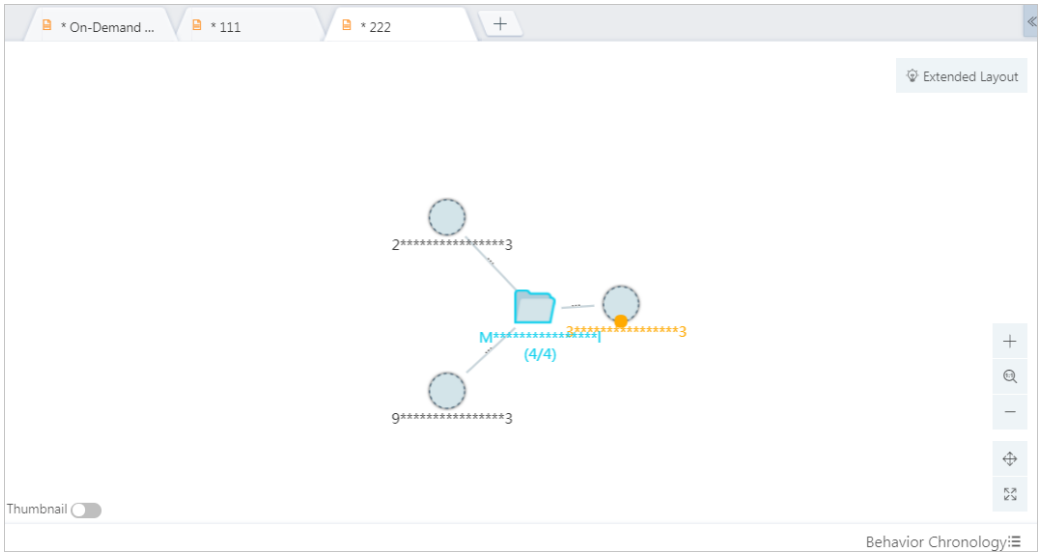
Matrix layout



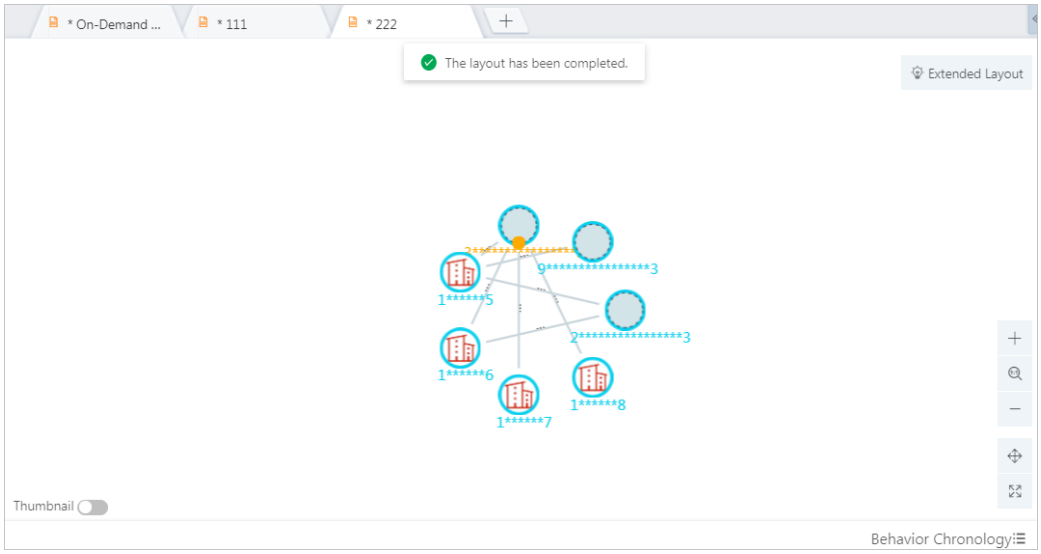
Merge nodes



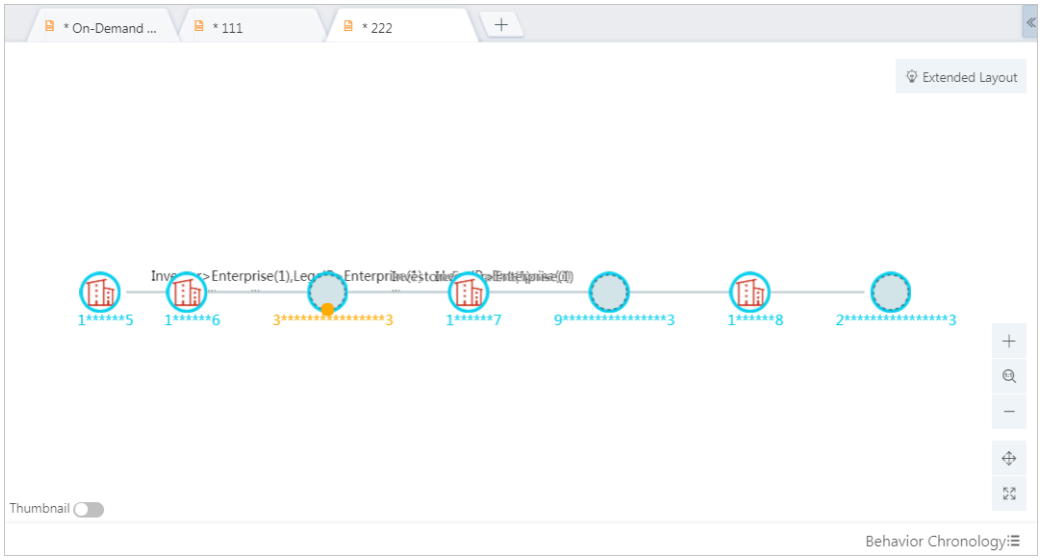
Merged result



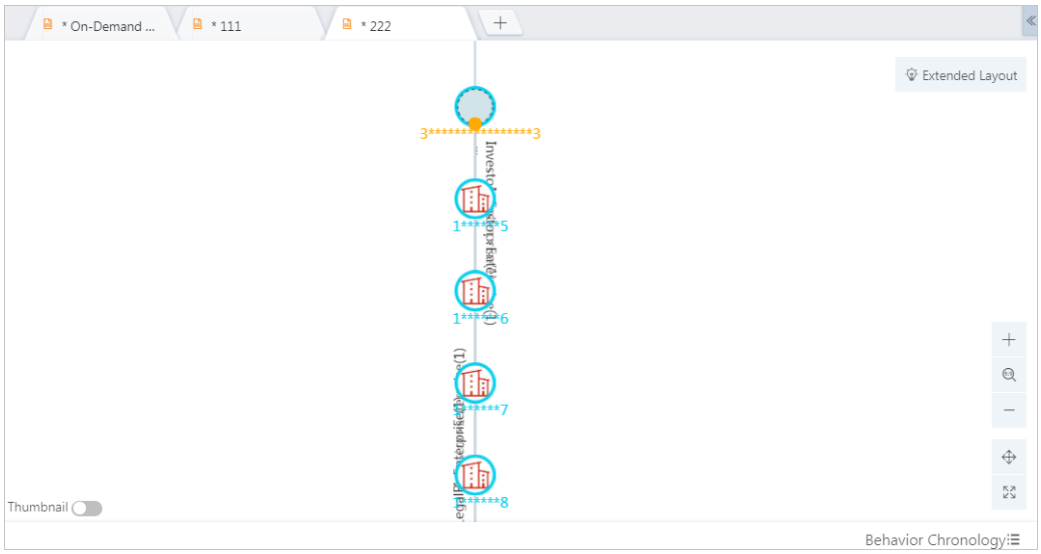
Circle layout



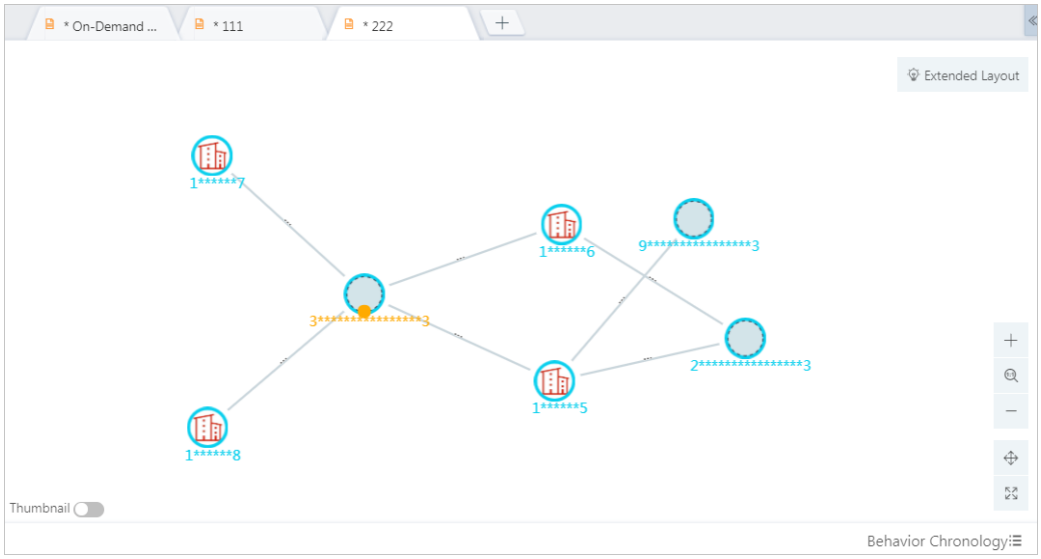
Horizontal layout



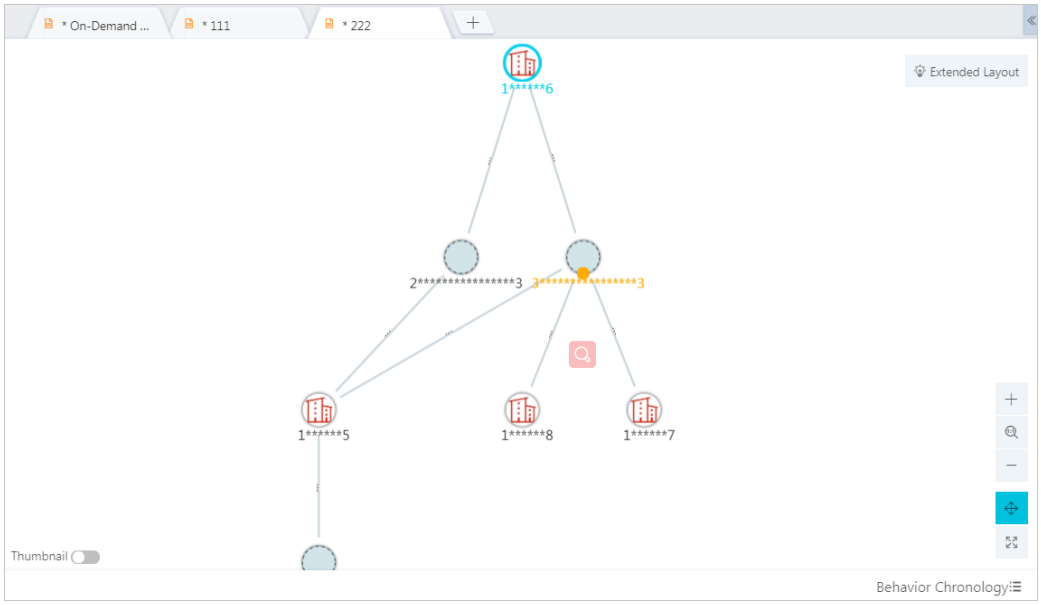
Vertical layout



Force-directed layout



Hierarchical layout



8.12.13. Flag and unflag nodes

When you analyze a large number of object nodes, you can highlight the key object node by adding a small red flag to the node.

Prerequisites

An object node already exists. For more information about how to add a node, see Add a node.

Context

You can perform flag operations on single nodes and merged nodes.

Procedure

1. Log on to Analytics Workbench.
2. Click an existing analysis file to open the file on the Graph page, or click **Create Analysis** to create an analysis file and add nodes.
3. Perform the flag and unflag operations as follows.

Operation	Procedure
Flag	Select the target nodes or merged nodes, and then click **Flag** in the top navigation bar. A red flag is displayed on the selected node or merged node.
Unflag	Select the target nodes or merged nodes, and then click **Unflag** in the top navigation bar. The red flag displayed on the selected node or merged nodes disappears.

8.12.14. Labels

8.12.14.1. Label types

You can use labels to identify the content, category, and other properties of the node, which is easy for you and other users to search and locate nodes. In Graph Analytics, you can add label content to the node objects in the graph area. Labels are divided into system labels and user labels.

You can attach a system label or a user label to each node object.

- **System labels**: labels attached to the node objects by the system based on algorithms.

  The system labels are displayed on the left side of the nodes. You cannot delete or modify the labels or click the Like button on the system label.

- **User labels**: labels attached to objects by the user.

  The user labels are displayed on the right side of the nodes. The user labels with certain permissions can be deleted and liked by corresponding users.

8.12.14.2. User labels

Graph Analytics supports four types of user labels. The visible ranges and operations for labels vary by type.

User label types

- **Public**

  After the analysis is shared, all people can see the label and click the Like button on this label.

- **Only Me**

  This label is visible only to the person who added the label. After the analysis is shared, other people cannot see this label.

- **Only for This Analysis**

  This label is visible only in the current analysis, and the Like button of the label can only be clicked in the current analysis. After a node with this label has been copied to another analysis, this label becomes invisible.

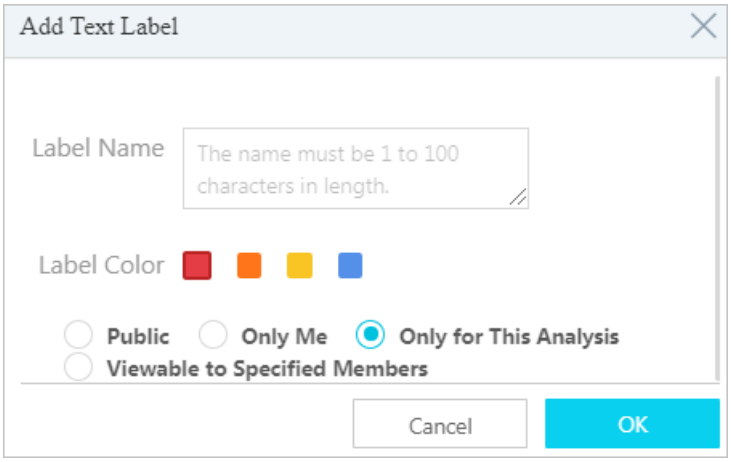
- **Viewable to Specified Members**

  This label can only be seen and liked by the users that are specified when the analysis is shared.

User label colors

You can add user labels of different colors. By default, four colors are provided to differentiate the labels: red, yellow, blue, and green, as shown in User label colors.

User label colors



8.12.14.3. Add user labels

When some nodes are hard to understand, you can classify and describe these nodes by adding labels. This helps you and other users to understand these nodes quickly and easily.

Prerequisites

Make sure that you have obtained an account and a password with the permission to perform graphic operations.

Context

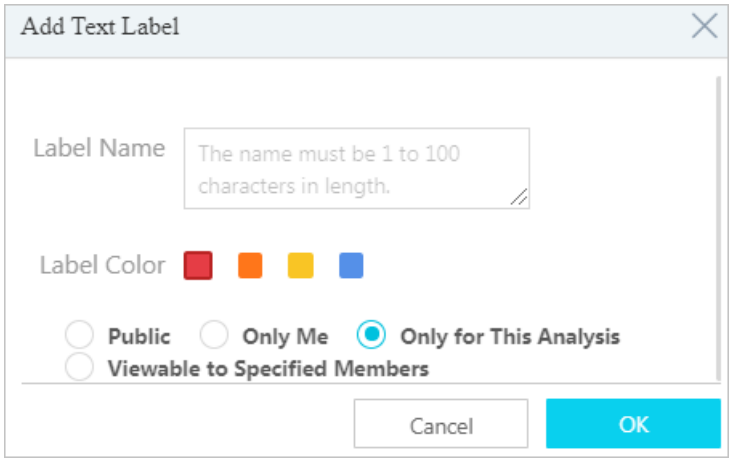
Based on the visible range, labels can be divided into four types: Public, Only Me, Only for This Analysis, and Viewable to Specified Members.

> **Note** You cannot add labels to merged nodes.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create an analysis.
3. You can use one of the following methods to add a label:
   - Select one or more nodes in the graph area, right-click a selected node, and then select **Add Label**. Set the parameters in the dialog box that appears.
   - Select one or more nodes on the **Details** tab of the right-side pane, click the **Text Note** icon in the top of the pane, and then set the parameters in the dialog box that appears.

   The parameters are described in Parameters and descriptions.



Parameters and descriptions

Parameter	Description
Label Name	The note or description of the node, which helps you and other users understand the node. The label name must be from 1 to 20 characters in length.
Label Color	The color of the label displayed in the graph area for easy identification. Four colors are supported: red, orange, yellow, and blue.
Visible range options	◦ **Public**: Users who use the node can see the label and click the Like button of the label. ◦ **Only Me**: The label is visible only to the person who added the label. Other users cannot see the label when they add this node to the graph area. ◦ **Only for This Analysis**: You can see the label and click the Like button of the label only in the current analysis. This label is invisible after this node is added to or appears in another analysis. ◦ **Viewable to Specified Members**: Only the specified users can see the label and click the Like button of the label. When you select **Viewable to Specified Members**, you must specify users as needed.

4. Click **OK**. A success message is displayed after the label has been added.

8.12.14.4. View labels

In Graph Analytics, you can view labels by type and easily analyze the object nodes.

Prerequisites

There are labels visible to you in the current analysis file.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis. You can switch the display mode of the labels based on your needs.

Display mode	Operation
All Labels	In the toolbar, click **All Labels**. The system labels of each node and the labels visible to you in the current analysis are displayed in the graph area.
My Labels	In the toolbar, click the label icon and select **My Labels**. All labels created by you for the displayed node are shown in the graph area.
Hide Labels	In the toolbar, click the label icon and select **Hide Labels**. All labels in the graph area are hidden.

8.12.14.5. Click likes and delete likes

When you agree on a label that was added by another user to a node, you can click the Like button for the label.

Prerequisites

- Make sure that you have obtained an account and a password with the permission to perform label-related operations.
- Make sure that other users have added labels that are visible to you.

Context

Only labels that are Public, Only Me, and Viewable to Specified Members can be liked. By default, every label has one like. When the last like of a label is canceled, the label is removed from the node.

After you like a label, the number of likes increases by 1, and the Like button changes to the Undo Like button. After you undo a like, the number of likes decreases by 1.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create an analysis, and add or locate a node that has user labels that are visible to you.
3. Click the label number next to the node, and click a label name. The detailed information about the specified label is displayed.

4. If you agree on the label content, click the Like button . A success message is displayed after the operation is completed.

What's next

If you think that a label you previously liked does not match the node, you can click the Undo Like button to undo the like.

8.12.14.6. Edit user labels

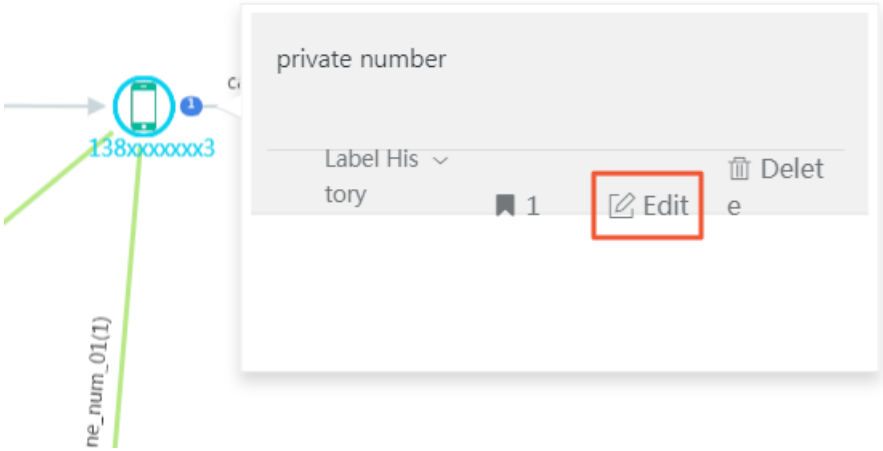
If the name, color, or visible range of a user label is unacceptable, you can edit the user label.

Prerequisites

- You have obtained the account and password with the permission to perform label-related operations.
- Other users have created user labels that are visible to you.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create an analysis, and add or locate a node that has user labels that are visible to you.
3. Click the label number next to the node, click a label name, and view the label details.



4. Click **Edit**, and re-set the parameters in the dialog box that appears.

   For more information about the parameter settings, see Parameters and descriptions in Add user labels.

5. Click **OK**.

8.12.14.7. Delete user labels

You can delete unnecessary user labels.

Prerequisites

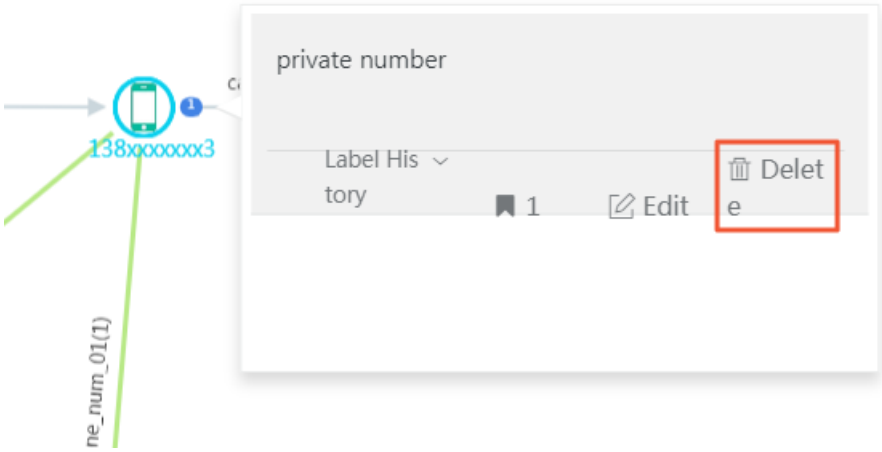
You have obtained the account and password with the permission to perform label-related operations.

Context

If the label is created by you and the number of likes is 1, you can cancel the like, and the number of likes becomes 0. When the number of likes for a label becomes 0, the system automatically deletes this label.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create an analysis, and add or locate a node that has user labels that are visible to you.
3. Click the label number next to the node, and click a label name to view the detailed information about the label.



4. Click the **Delete** icon, and click **OK** in the dialog box that appears. A message is displayed, indicating that the label has been deleted successfully. If the label is created by you and currently has only one like, you can click the delete icon to delete the like, and the number of likes becomes 0. The system will automatically delete this label.

8.12.15. Save analysis

After the analysis is modified, you must save the modifications before you close the analysis. Graph Analytics provides a screenshot analysis function. When an analysis is saved, Graph Analytics generates a global screenshot for the analysis graph and saves the screenshot to the server.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created a new analysis or modified an existing analysis.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis file or create a new analysis.
3. (Optional)Perform multiple operations on an existing analysis or a new analysis in the graph area, such as adding nodes, setting layouts, and performing analyses.
4. The following may occur when you click the **Save** icon in the toolbar.

Operation	Description
Save existing analyses	Saves the analysis content in the graph area by the original name and the original path.
Save new analyses	If you are creating a new analysis, you need to set File Name and Folder of the analysis in the **Save Analysis** dialog box that appears, and click **OK** to save the analysis content in the graph area.

8.12.16. Print graph areas

Analytics Workbench enables you to print the content of an analysis from different perspectives. You can export the content as a paper document, or save the content as a PNG image.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- Make sure that the system has connected to a printing device.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis, or create a new analysis and have the results analyzed.
3. You can print the analysis content or save the analysis content as an image.

Print area	Operation

Print area	Operation
Print the full graph	i. In the toolbar, choose **Print** > **Print Full Graph** . The print settings page appears. ii. Set the print parameters, and then click **Print** to print all the analysis content in the graph area.
Print the visible area	i. In the toolbar, choose **Print** > **Print Visible Area** . The print settings page appears. ii. Set the print parameters, and then click **Print** to print the analysis content in the visible area.
Print the selected subgraph	i. In the toolbar, choose **Print** > **Print Selected** . The print settings page appears. ii. Set the print parameters, and then click **Print** to print the analysis content of the selected nodes in the graph area.
Save the full graph as an image	In the toolbar, choose **Print** > **Save Full Graph as Image** . Save the analysis content as an image in the PNG format.

8.12.17. Share analyses

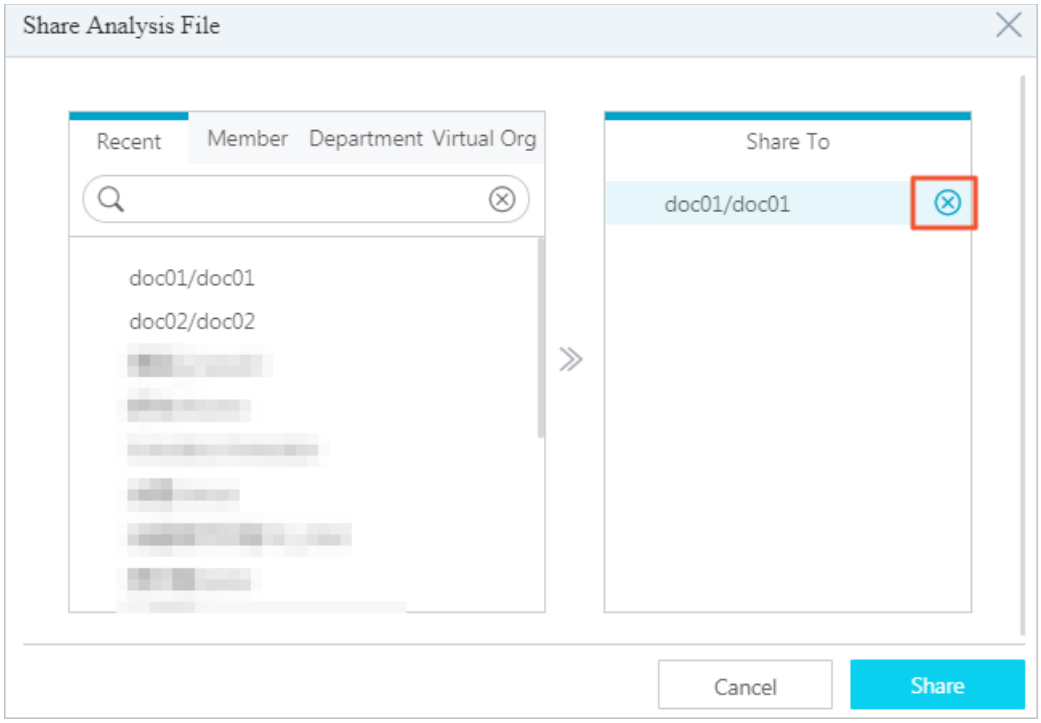
You can use Graph Analytics to share your current analysis with a specific user or group to perform a collaborative analysis. The shared users can view the shared analysis file after they log on to Analytics Workbench.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Analytics Workbench.
2. Open an existing analysis or create a new analysis.
3. In the toolbar, click the **Share** icon and set the parameters in the dialog box that appears.



You can select the shared members individually by using the search and positioning feature. You can also select the shared members by department.

Members selected for sharing will be displayed in the **Share To** list on the right side. When the mouse pointer is moved over the member, you can see a Delete icon. You can click the icon to delete the current member.

4. After you have specified the shared members, click **Share**, and the system informs you that you have shared the analysis successfully.

What's next

The user you shared files with can choose **File Center** > **Shared with Me** in the top navigation bar to view and operate on the shared analysis files.

After a member receives a shared analysis, the system automatically creates a directory with the same name as the source analysis on the **Shared with Me** page. By default, the directory has two files: the **initial file** and the **automatically merged file**.

8.12.18. Behavior chronology

8.12.18.1. Details

Details are used to display the link and event details of an object.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

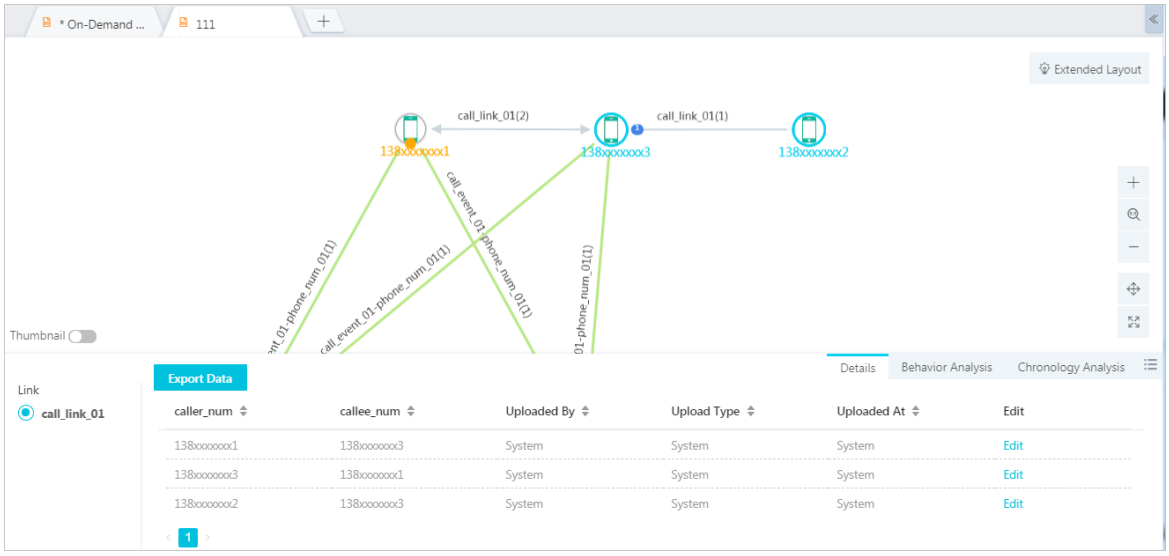
Procedure

1. Log on to Analytics Workbench.

2. Click an analysis file to open the file in the **Graph** area.
3. You can open the **Behavior Chronology** pane by using the following methods.

Method	Procedure
Method 1	Select an object in the **Graph** area, and click **Behavior Chronology** in the lower-right corner. The **Details** area is displayed by default.
Method 2	Select a link in the **Graph** area, and the **Behavior Chronology** pane appears automatically. The **Details** area is displayed by default.

4. On the left side of the **Behavior Chronology** pane, select a link or an event to view its details.



5. (Optional)Click **Export** to export the lists of behavior details for all links to an Excel file.

8.12.18.2. Behavior analysis

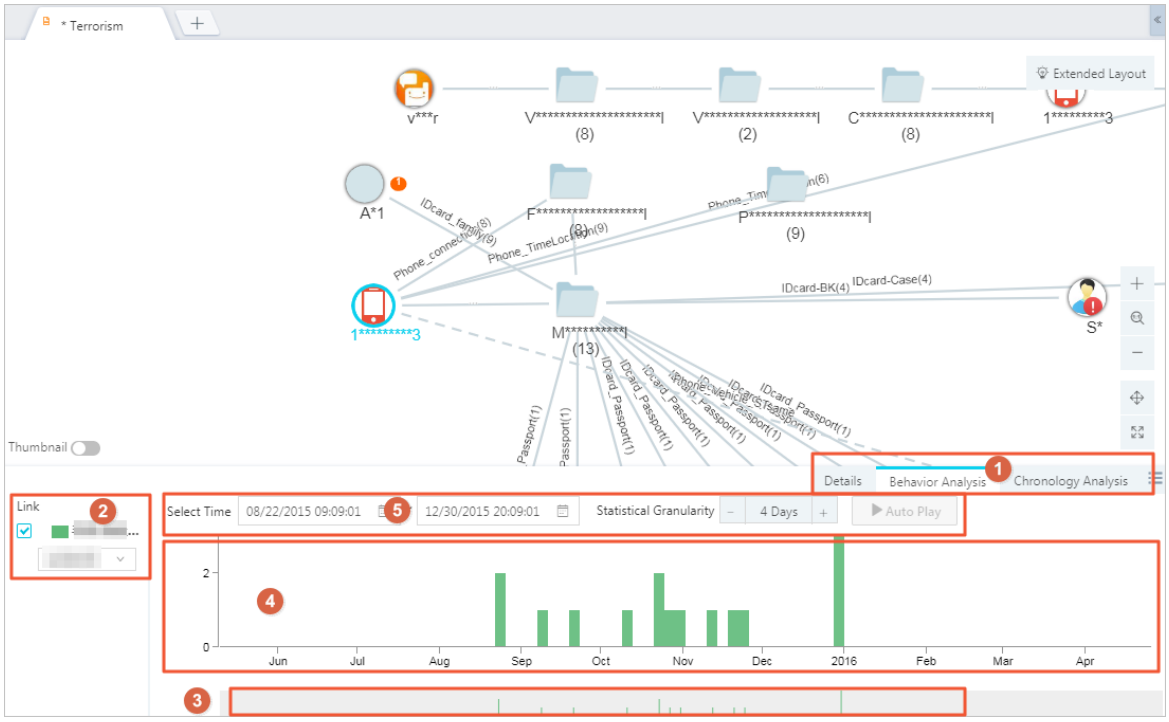
This feature allows you to display and analyze the link data with time properties on the Graph page.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

Procedure

1. Log on to Analytics Workbench.
2. Click an analysis file to open the file in the **Graph** area.
3. In the **Graph** area that appears, select an object for the behavior analysis.
4. Click the **Behavior Chronology** icon in the lower-right corner. In the **Behavior Chronology** pane that appears, click the **Behavior Analysis** tab.



The behavior analysis area is described as follows.

Description of the behavior analysis area

Area	Description
Area 1	You can switch to the **Behavior Analysis** tab.
Area 2	You can filter the links to be analyzed.
Area 3	You can filter data using the thumbnail. You can locate the range through the thumbnail at the bottom, and select the middle part to move the thumbnail.
Area 4	Move your mouse pointer to the column chart. Links that occurred at the current time point and the number of times the links have occurred are displayed. Click the column chart. The links and objects correlated to these links will be highlighted in the **Graph** area. Click and hold down the left mouse button and drag to select a time range. All links in the time range and objects correlated to these links will be highlighted in the Graph area.
Area 5	You can set the filter time range and the statistical granularity: ○ Set the start time and the end time for time-based filtering. ○ Click the + icon or the - icon to increase or decrease the statistical granularity.

5. For more information about how to analyze the behaviors of an object in the **Behavior**

Analysis area, see Description of the behavior analysis area.

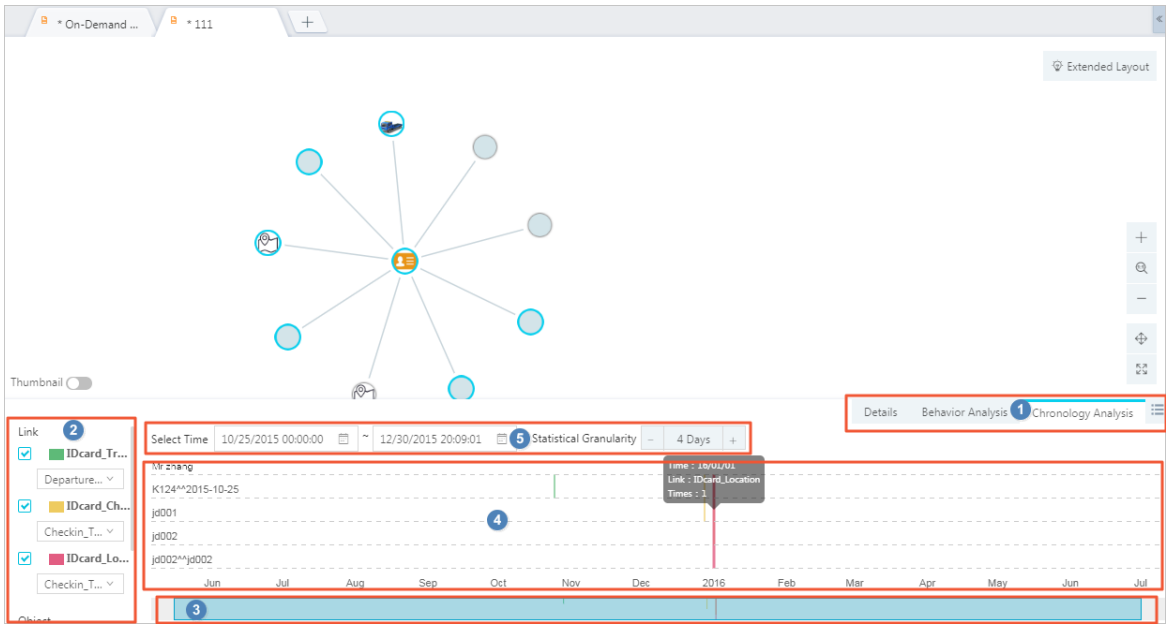
8.12.18.3. Chronology analysis

The chronology analysis shows the details of each event based on time.

Prerequisites

Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

1. Log on to Analytics Workbench.
2. Click an analysis file to open the file in the **Graph** area.
3. In the **Graph** area, select the object and link to perform a chronology analysis.
4. Click the **Behavior Chronology** icon in the lower-right corner. In the **Behavior Chronology** pane that appears, click the **Chronology Analysis** tab.



By default, only five objects can be displayed for an analysis in the chronology analysis area.

The chronology analysis area is described as follows:

Description of the chronology analysis area

Area	Description
Area 1	You can switch to the **Chronology Analysis** tab.
Area 2	You can select a link and an object. You can analyze a maximum of five objects at the same time.
Area 3	You can filter data using the thumbnail. You can locate the range through the thumbnail at the bottom, and select the middle part to move the thumbnail.

Area	Description
Area 4	When you move the mouse pointer to a specific chronology line, the details of the chronology appear. When you click the chronology line, the links and the objects involved in these links will be highlighted in the **Graph** area. Click and hold down the left mouse button and drag to select a time range. All links in the time range and objects correlated to these links will be highlighted in the **Graph** area.
Area 5	You can set the filter time range and the statistical granularity: ○ Set the start time and the end time for time-based filtering. ○ Click the + icon or the - icon to increase or decrease the statistical granularity.

5. For more information about how to analyze the behaviors of an object in the **Behavior Analysis** area, see Description of the chronology analysis area.

8.12.19. Property statistics

8.12.19.1. Details

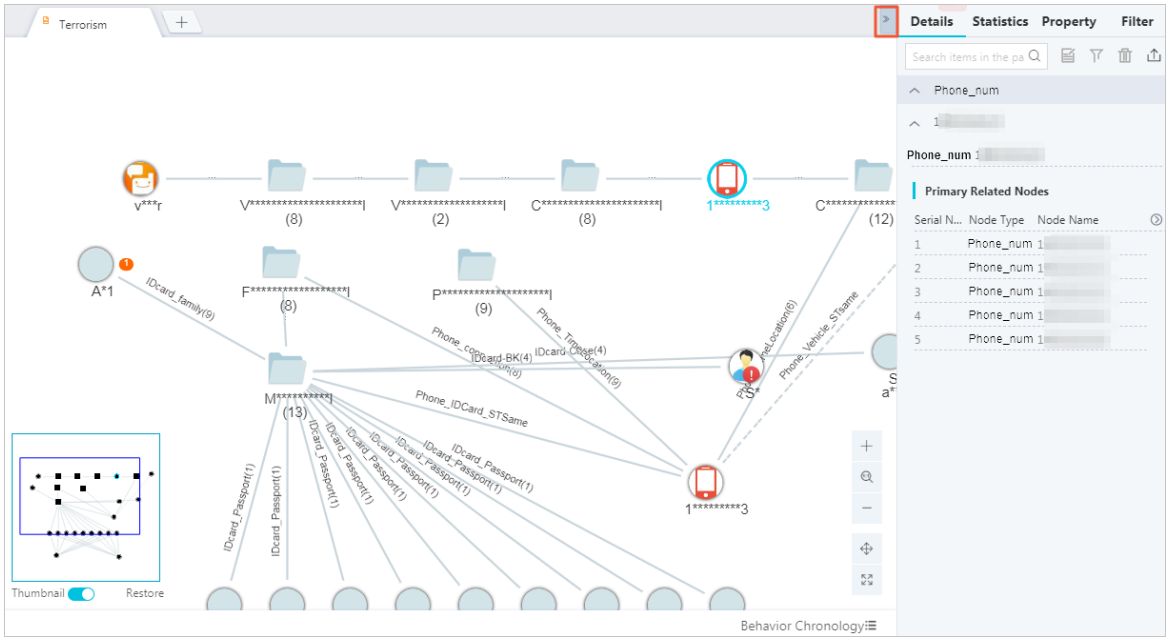
Typically, when you perform an analysis in Graph Analytics, a large number of objects will be involved. You may need to highlight the objects that are important to you. The properties and statistics area displays data with multiple properties. You can highlight the key objects on the Graph page.

Prerequisites

You have obtained an account and a password with the permission to perform network analyses.

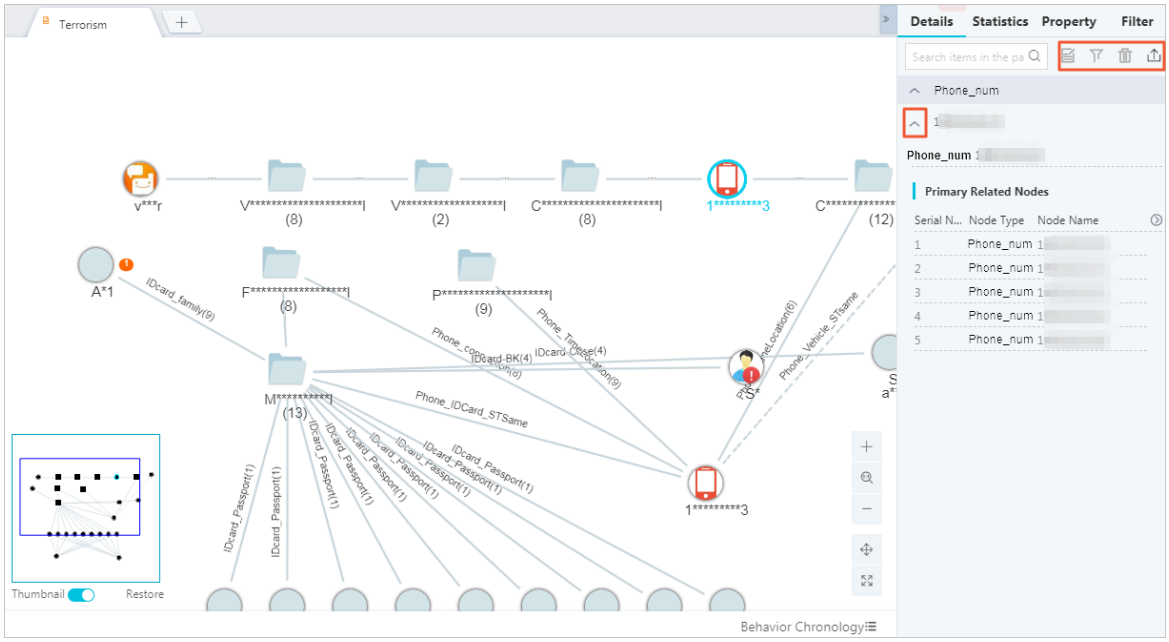
Procedure

1. Log on to Analytics Workbench.
2. Click an analysis file to open the file in the **Graph** area.
3. Select one or more objects or links in the **Graph** area.
4. Click the icon in the right side of the graph area to display the properties and statistics. By default, the **Details** tab is displayed.



The **Details** tab displays the basic information of the selected object, including the object type, object icon or avatar, object ID, object properties, and the correlated nodes.

5. You can perform the following operations in the **Details** tab.



Operation	Procedure
Highlight key objects	When you select objects in the right-side pane, the objects, links, and events in the graph area are masked, except for the selected objects. You can press the **Control** key to select multiple objects.

Operation	Procedure
View object details	When you select nodes or links in the main graph area, the details page on the right side shows all the property information of the selected nodes and links. You can press the **Control** key to select multiple nodes or links. If you select a link in the graph area, the details of the objects involved in the link are displayed.
Add text notes	Add text notes or labels. For more information, see Add user labels.
Select	Select the node in the graph area.
Delete	Delete the selected object in the graph area.
Export	Export the information of the selected object to an Excel file.

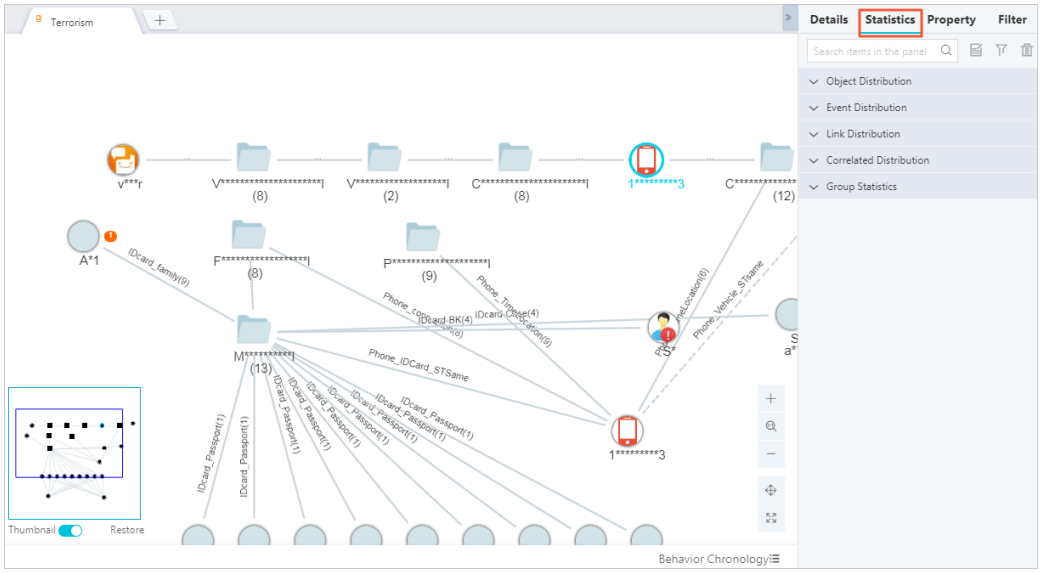
8.12.19.2. Statistics

The Statistics tab displays the statistics of the selected objects or links.

Procedure

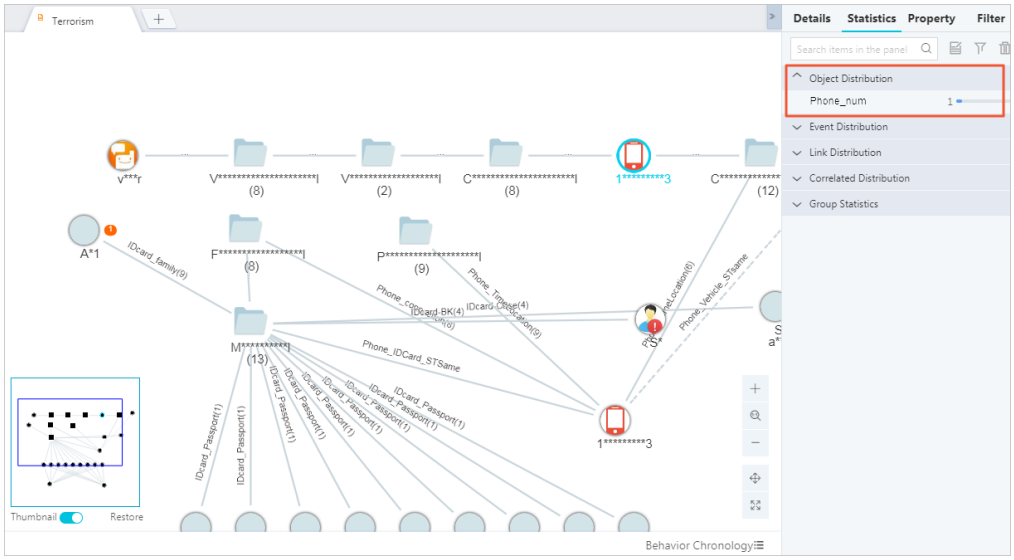
1. Log on to Analytics Workbench.
2. Select one or more nodes or links in the graph area.
3. Click the right arrow in the right side of the graph area to display the properties and statistics.
4. Switch to the **Statistics** tab, as shown in Statistics tab.

   Statistics tab



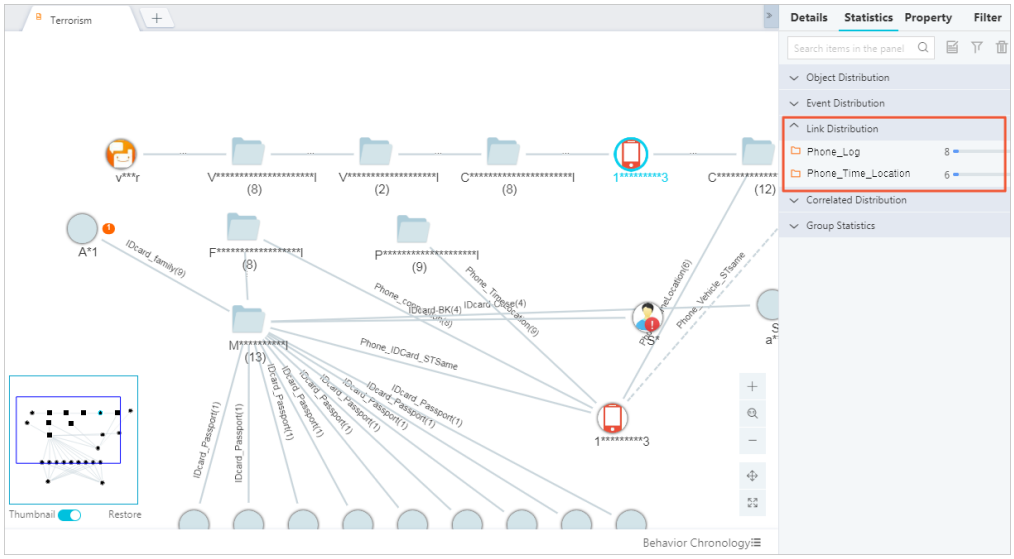
5. Click the up or down arrow to hide or show the properties of the selected object or link.
   - View the object distributions, as shown in Object distributions.

Object distributions



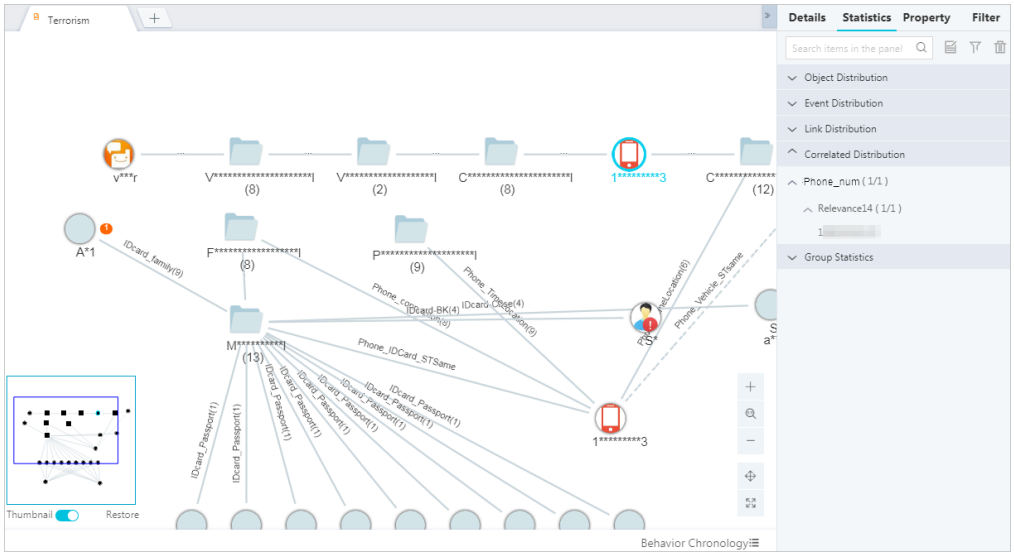
- View the link distributions, as shown in Link distributions.

  Link distributions



- View correlation distributions, as shown in Correlation distributions.

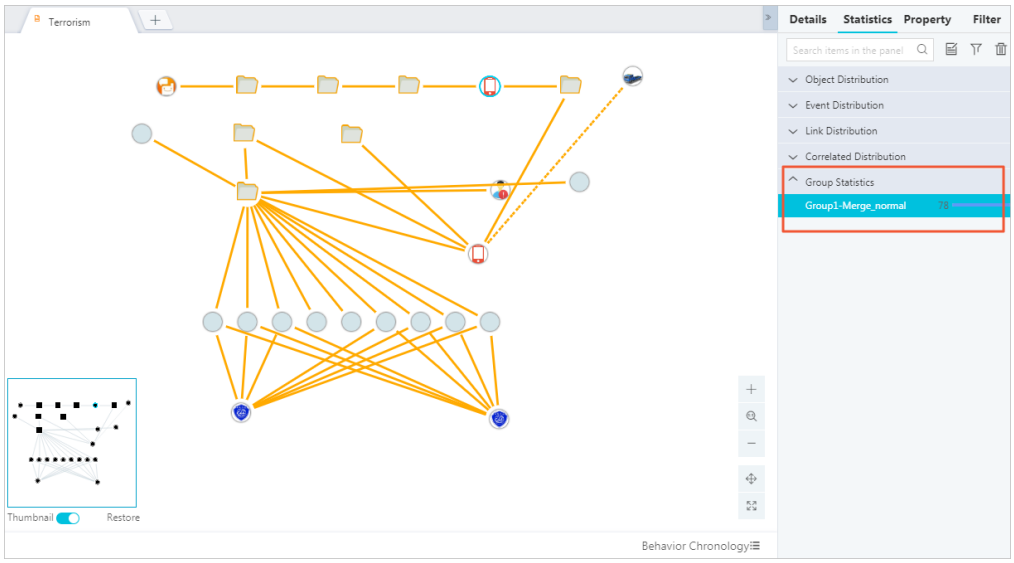
  Correlation distributions



- View the group analysis, as shown in Group analysis.

  Group statistics allows you to analyze group distributions in Graph Analytics. Group statistics is typically used to perform a group analysis. After you enter multiple nodes, Graph Analytics analyzes the interrelations between these nodes, provides the analysis results, and displays the group distribution. A group consists of multiple object nodes, with any two object nodes connected topologically. Nodes within a merged node are connected topologically.

  Group analysis



  - In group statistics, all isolated nodes form a group are called isolated nodes.
  - The list in group statistics displays the number of nodes and the labels of the nodes that have the highest correlated degree in each group, excluding the group of isolated nodes. Group of isolated nodes is displayed at the top of the list, while other groups are listed in a descending order according to the number of nodes contained in the group.

> **Note** Group statistics cover all nodes in Graph Analytics, regardless of your selection of nodes during the analysis process.

On the Statistics page, you can perform the following operations for the selected content:

- Text Note: You can add text notes. For more information, see Add user labels.
- Selected: You can select the node in the graph area.
- Delete: You can delete the selected object in the graph area.

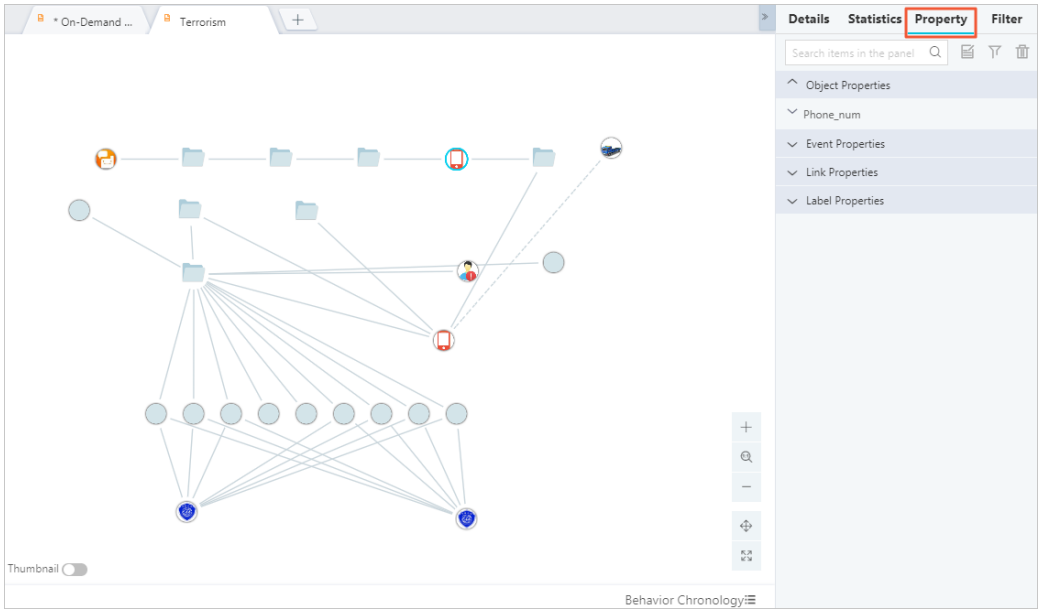
8.12.19.3. Property information

The Property tab in the right-side pane of the Graph page displays the object or link information and the label information of the selected objects or links.

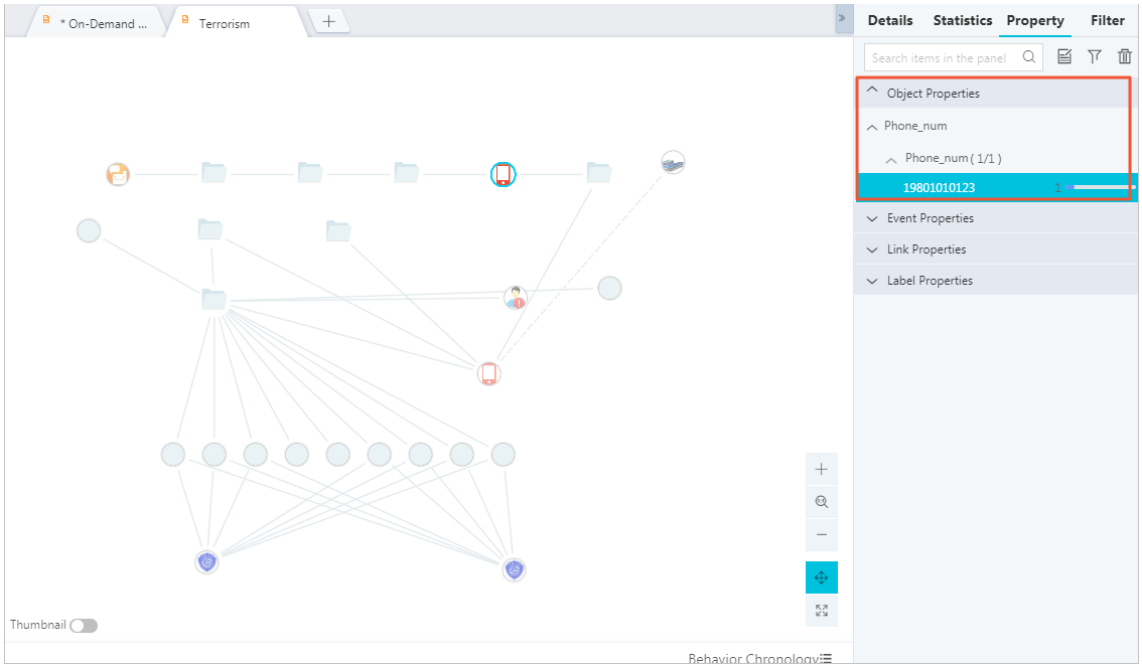
Procedure

1. Log on to Analytics Workbench.
2. Select one or more nodes or links in the graph area.
3. Click the right arrow in the right side of the graph area to display the properties and statistics.
4. Click the **Property** tab. The Property tab appears, as shown in Property tab.

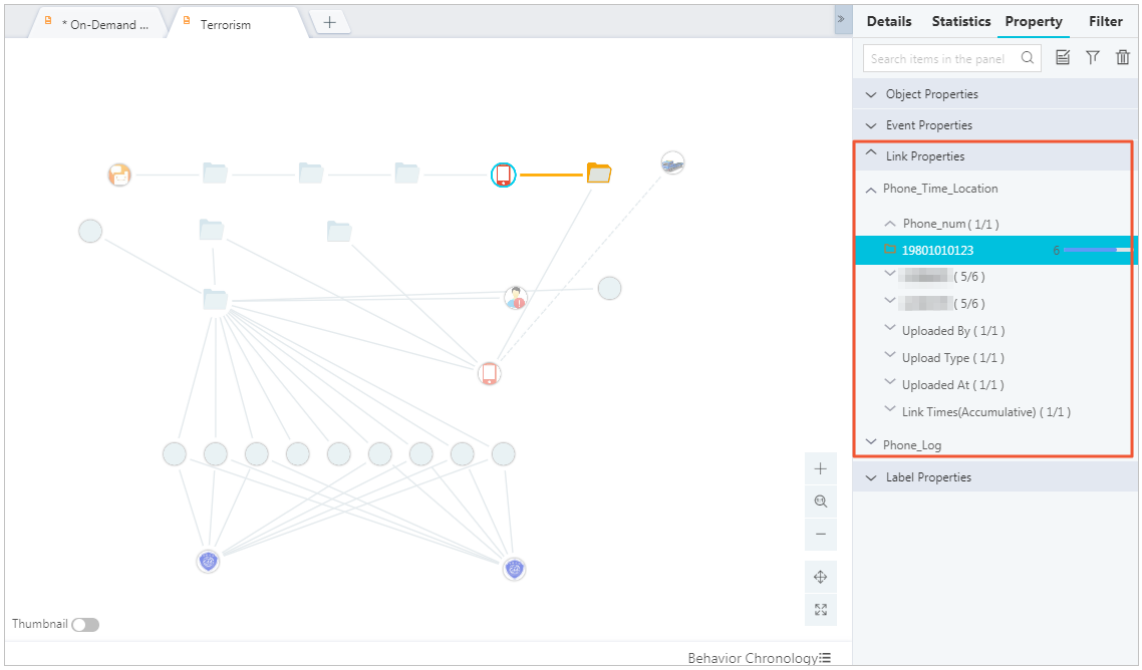
   Property tab



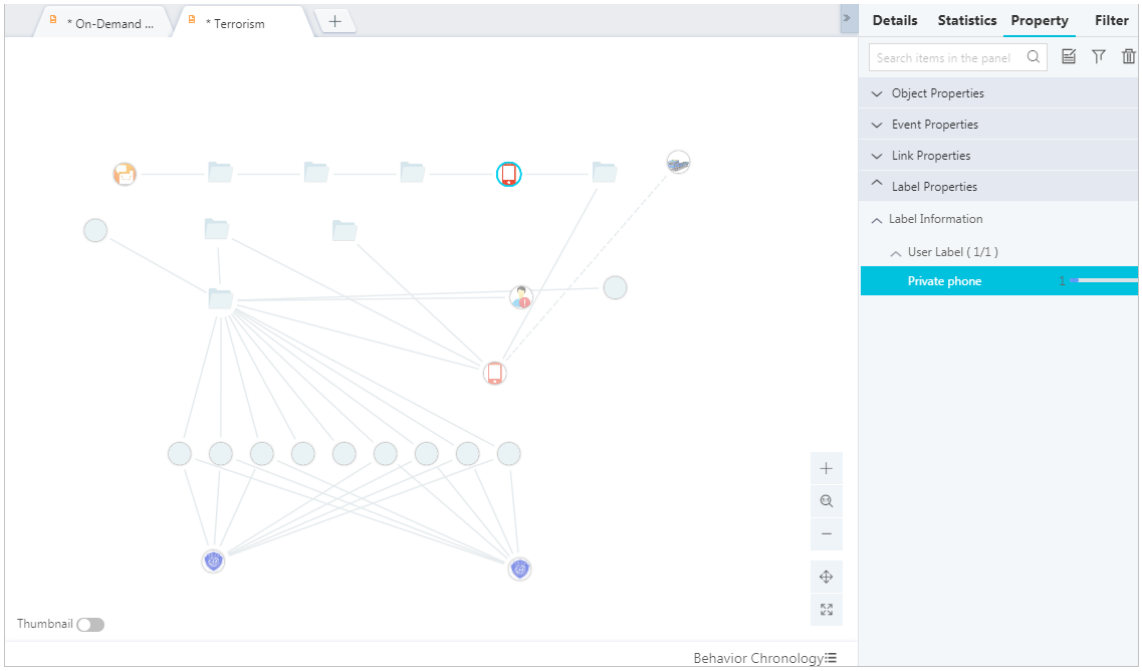
5. Click the up or down arrow to hide or show the properties of the selected object or link.
   - View object properties.



- View event properties.
- View link properties.



- View label properties.



On the Property tab, you can perform the following operations for the selected content:

- Text Note: You can add text notes. For more information, see Add user labels.
- Selected: You can select the node in the graph area.
- Delete: You can delete the selected object in the graph area.

8.12.19.4. Secondary filtering

You can use the secondary filtering feature to filter objects, links, or events in the canvas. In a complex analysis, you can filter out irrelevant objects, links, or events to keep the content simple and concise.

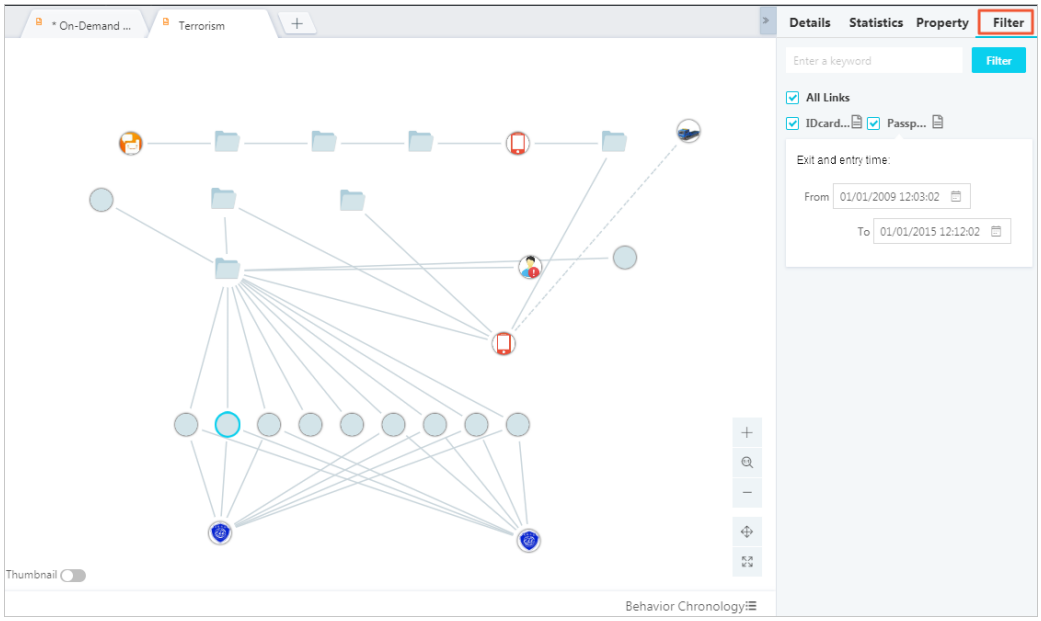
Context

In many service scenarios where data processing is complex, the canvas analysis may become very complex after a few analysis extensions. It is essential to perform secondary filtering before you make further analyses and judgments.

Procedure

1. Log on to Analytics Workbench.
2. Click the arrow-pointing-right icon in the right of the graph area to show the properties and statistics.
3. Click the **Filter** tab, and switch to the **Filter** page, as shown in Entry to the secondary filtering.

   Entry to the secondary filtering



4. Enter a keyword to be filtered.

   Related parameters are described as follows:

   - Time type: maximum and minimum values of the time property. You can adjust the value range, for example, departure time.
   - Numeric type: maximum and minimum numerical values. You can adjust the value range, for example, age.
   - Dictionary type: enumerated values. You can delete some of the enumerated values.
   - Character string: used for searching. Fuzzy search is supported.

8.13. File Center

8.13.1. View and manage all analyses

You can see all the personal analysis files and shared analysis files of the current user. The files are arranged in the order of creation time.

Prerequisites

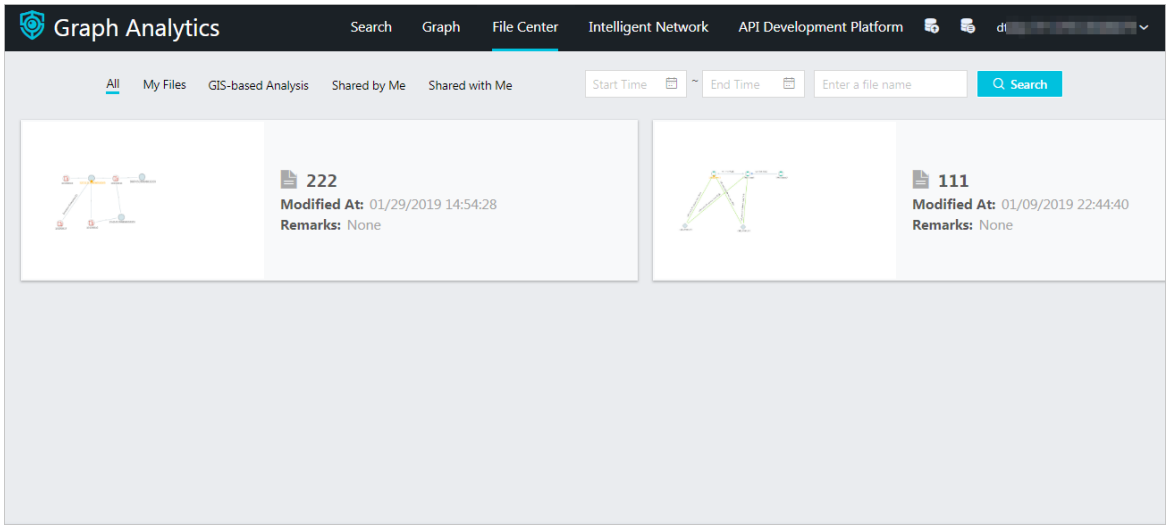
The current user has created and saved analysis files, or has received shared analysis files.

Procedures

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center**, and then click the **All** tab. The **All** tab appears.

   The **All** page displays all the shared files and personal files.

   When you handle a large number of analysis files, you can use the Search feature to find the target files.



3. On the **All** page, you can perform the following operations on each analysis:

Method	Description
Open an analysis file	Double-click an analysis file to open the file in the graph area.
Delete an analysis file	Select an analysis that you created or a shared file that you received, click the **Delete** icon in the lower-left corner, and click **Delete** in the dialog box that appears.
Rename an analysis file	**Note** This operation is only valid for analysis files created by you. Select an analysis, and click the **Rename** icon in the lower-left corner. In the dialog box that appears, enter a new name and click **OK**.
Change sharing permissions	**Note** This operation is only valid for files that have been shared by the current user. Select a shared file, and click the **Change Sharing Permissions** icon in the lower-left corner. In the dialog box that appears, reset the shared members and click **Share**.

8.13.2. View and manage your files

You can view all your personal folders and analysis files in the order of creation time. You can perform operations on the folders and analysis files, such as add, delete, view, and modify.

Prerequisites

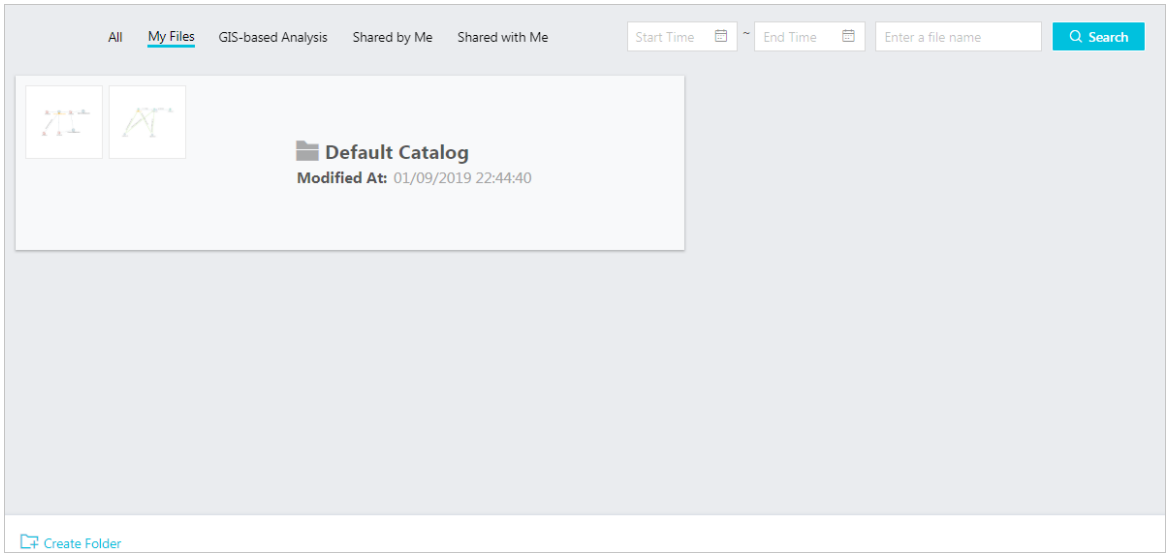
Make sure that you have created and saved an analysis file.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center** and select the **My Files** tab. The **My Files** page appears.

   The **My Files** page displays all folders that contain analysis files that have been saved.

   You can use the Search feature to handle a large number of folders or analysis files.



3. On the **My Files** page, you can perform the following operations.

Operation	Description
Create a folder	Click **Create Folder** in the lower-left corner of the page. In the dialog box that appears, enter a folder name, and then click **OK**.
Rename a folder	Select the folder to be renamed, and click **Rename** in the lower-left corner of the page. In the dialog box that appears, enter a new name, and then click **OK**.
Delete a folder	Select the folder to be deleted, and click **Delete** in the lower-left corner of the page. In the message that appears, click **Delete**.
Rename an analysis file	i. Double-click a folder to open the folder. ii. Select an analysis file to be renamed, and click **Rename** in the lower-left corner of the page. In the dialog box that appears, rename the analysis file, and then click **OK**.
Delete an analysis file	i. Double-click a folder to open the folder. ii. Select an analysis file to be deleted, and click **Delete** in the lower-left corner of the page. In the message that appears, click **Delete**.

Operation	Description
Move an analysis file	i. Double-click a folder to open the folder. ii. Select the analysis file to be moved, and click **Move** in the lower-left corner of the page. In the dialog box that appears, select the target folder, and click **OK**.
Share an analysis file	i. Double-click a folder to open the folder. ii. Select the analysis file to be shared, and click **Share** in the lower-left corner of the page. In the dialog box that appears, select the members you want to share the file with, and click **Share**.

8.13.3. My shared items

8.13.3.1. Overview

Graph Analytics allows you to share personal analyses with others. You can share personal ideas and experiences with other users and combine their intelligence and experiences to achieve collaboration and build a better team.

The initiator and the collaborators are the main roles in the collaboration and sharing process. A collaboration proceeds as follows.

After a member receives a shared analysis, the system automatically creates a directory with the same name as the source analysis on the **Shared with Me** page. By default, the directory has two files: the **initial file** and the **automatically merged file**.

Graph Analytics allows you to manage shared files, including deleting files, renaming files, and managing version history.

> **Note** Only initiators have the permissions to delete and rename shared files.

8.13.3.2. View and manage shared files

The Shared by Me page displays all the files shared by the current user in the order of time when the files were created. You can delete folders and change sharing permissions. You can also merge shared files and delete version files.

Prerequisites

You have shared an analysis file. For more information, see Share analyses.

Context

On the Shared by Me page, you can perform the following operations.

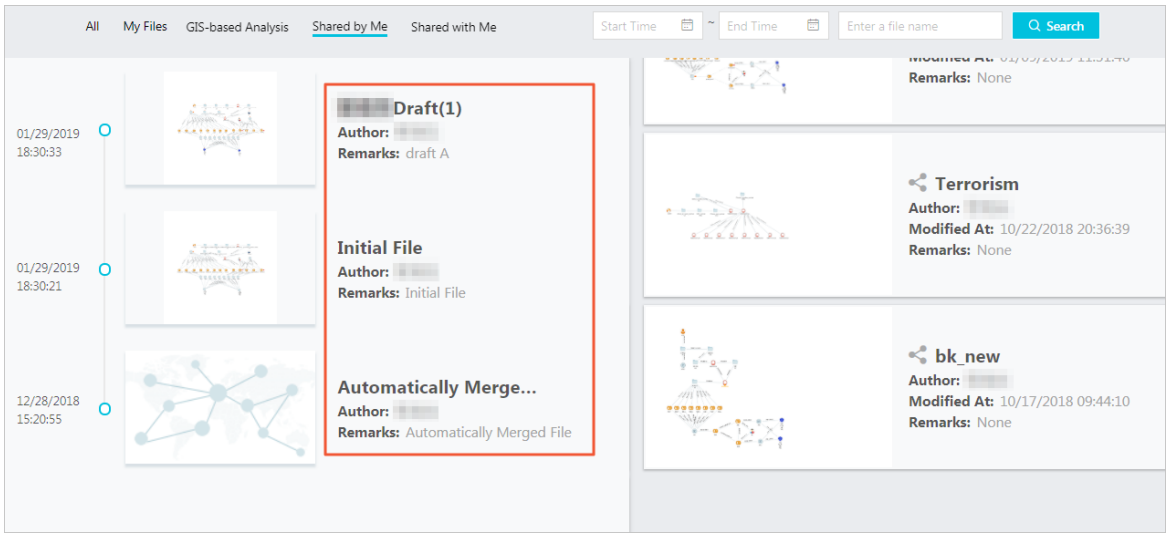
Operation	Description
Delete a shared folder	After the sharer deletes the shared folder, the files will also be deleted from members to which the files are shared.
Modify sharing permissions	After the sharer modifies the sharing permissions, the permissions of the original sharing members are revoked, and the shared files are also deleted.
Delete a shared file	Note You cannot delete the initial file or the automatically merged file that exist by default.
Manually merge files	You can merge multiple published versions in a shared folder to form a new version. Only published file versions can be merged. After the files are merged, a manually merged file is generated. If the published versions are different, use the union of the versions.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center**, and the click the **Shared by Me** tab. The **Shared by Me** page appears.

   The **Shared by Me** page displays all folders that the current user has shared. Each folder is a shared item. It contains multiple draft files, one initial file, one automatically merged file, multiple manually merged files, and multiple published version files.

   When a large number of folders or files are displayed, you can use the search function to query the target.



3. On the **Shared by Me** page, you can perform the following operations.

Operation	Description

Operation	Description
Delete a shared folder	Select a shared folder, click the **Delete** icon in the lower-left corner, and click **Delete** in the dialog box that appears.
Modify sharing permissions	Select a shared folder, and click the **Change Sharing Permissions** icon in the lower-left corner. In the dialog box that appears, reselect the members to whom you want to share the folder, and click **Share**.
Delete a shared file version	i. Double-click the shared folder, and select a file. You can delete draft files, manually merged files, and published version files. ii. Click the **Delete** icon in the lower-left corner, and click **Delete** in the dialog box that appears.
Manually merge files	Select two or more published version files, and click the **Merge Files** icon in the lower-left corner. In the dialog box that appears, enter the description and click **OK**.

8.13.3.3. Edit a shared file

The sharer and shared members can edit shared files, including initial files and automatically merged files. After a file is edited and saved, a draft file will be generated. Only the current user can view the draft.

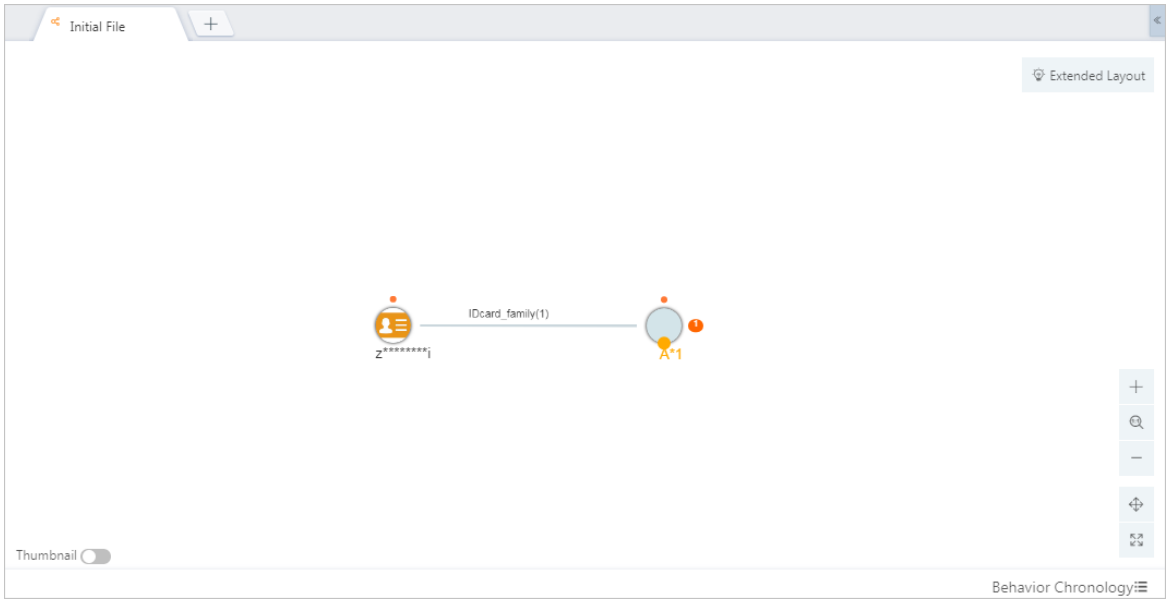
Prerequisites

You have shared an analysis file. For more information, see Share analyses.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center**, and then click the **Shared by Me** tab. The **Shared by Me** page appears.
3. Double-click a shared folder, and then double-click a shared file in the folder, such as the initial file or the automatically merged file, to open the shared file in the graph area.

   Each object node in the initial file has a red dot above it to identify itself as the initial node. Subsequently extended object nodes do not have this red dot.

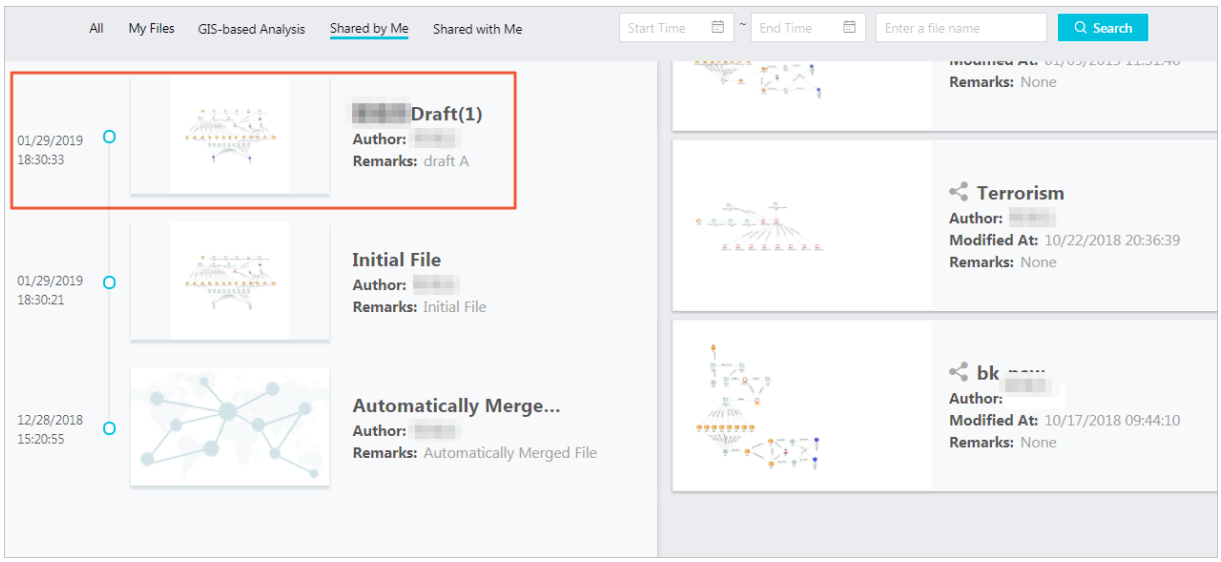


4. Perform analysis or layout operations on the shared file as needed.
5. After the analysis results are shown, click the **Save** icon in the toolbar. In the dialog box that appears, enter the Draft Statement and click **Save**.

   Draft Statement: Enter the draft statement. The statement must be 1 to 200 characters in length.

Result

After a draft of a shared file is saved, the draft file is displayed in the directory of the shared file. However, the draft is visible only to the current user.



- Location: above the **initial file**.
- Name format: username of the current user + draft + (number). For example, test123draft(1).
- Author: the user who saved this draft.
- Description: displayed below the draft file name.
- Time: the time when the draft was created is displayed on the left side of the file.

8.13.3.4. Publish a version

You can edit a shared analysis file and publish a new version. The published analysis file can be viewed by all users with whom the file is shared. However, only the sharer and the publisher can delete the file.

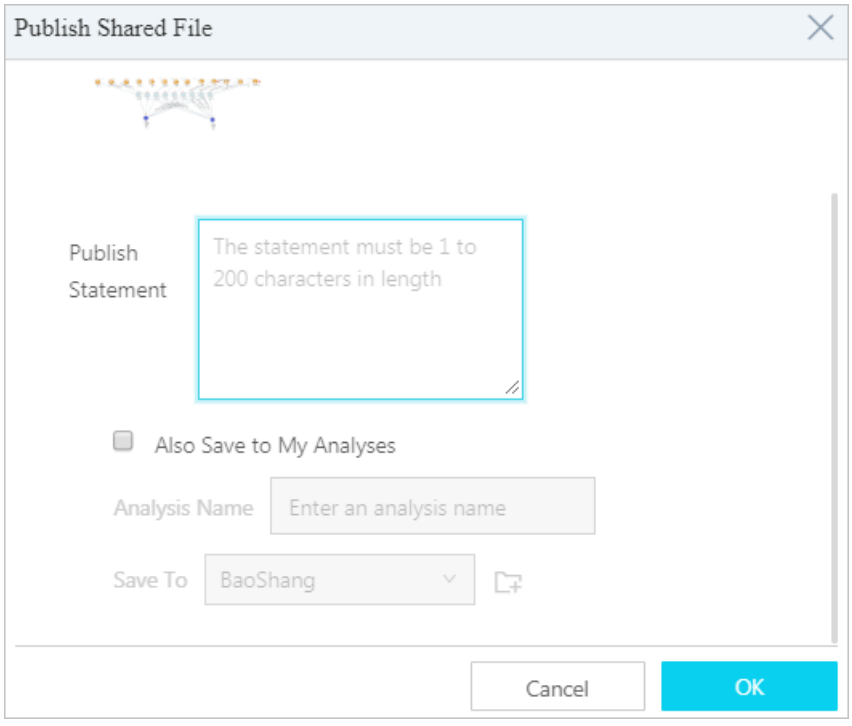
Prerequisites

You have shared an analysis file. For more information, see Share analyses.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center**, and then click the **Shared by Me** tab. The **Shared by Me** page appears.
3. Double-click a shared folder, and double-click a shared file in the folder, such as the initial file or the automatically merged file, to open the shared file in the graph area. Perform analysis or layout operations on the shared file as needed.
4. After the analysis results are shown, click the **Publish** icon in the toolbar and set the parameters in the dialog box that appears.

   The parameters are described in Publish parameters.



Publish parameters

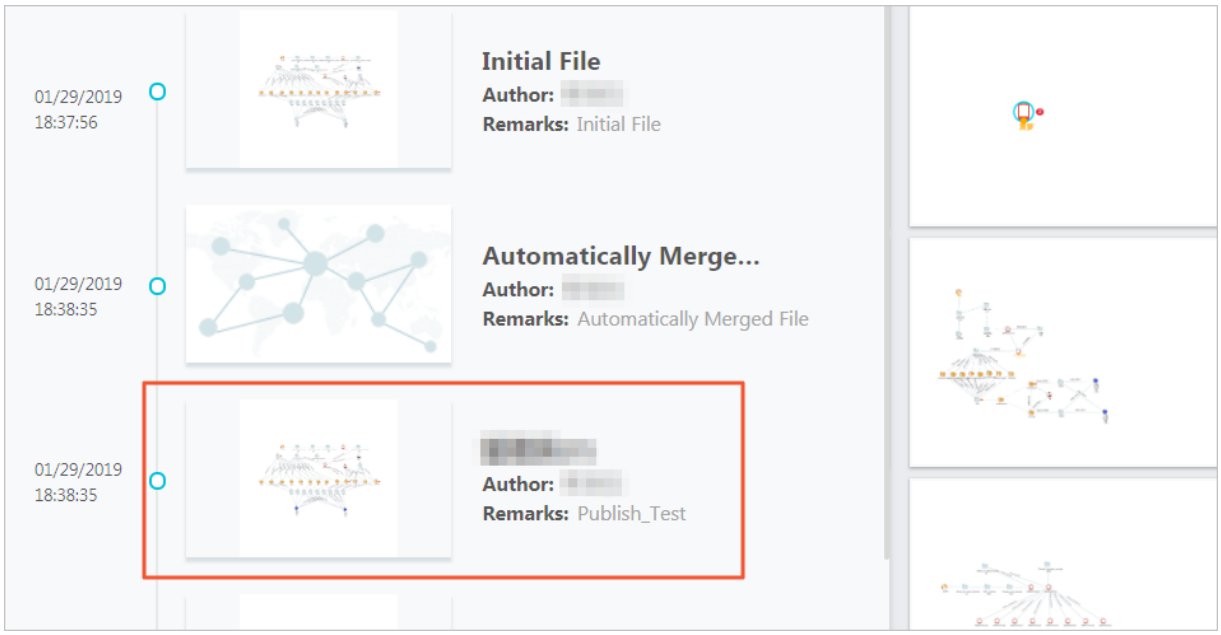
Parameter	Description
Publish Statement	Enter the publish statement. The statement must be 1 to 200 characters in length.

Parameter	Description
Also Save to My Files	After you select this checkbox, the analysis file is displayed on the My Files tab page. You must enter the analysis name and the directory of the analysis file.

5. Click OK.

Result

After a new version of a shared analysis is published, the published version file is displayed in the directory of the shared analysis file.



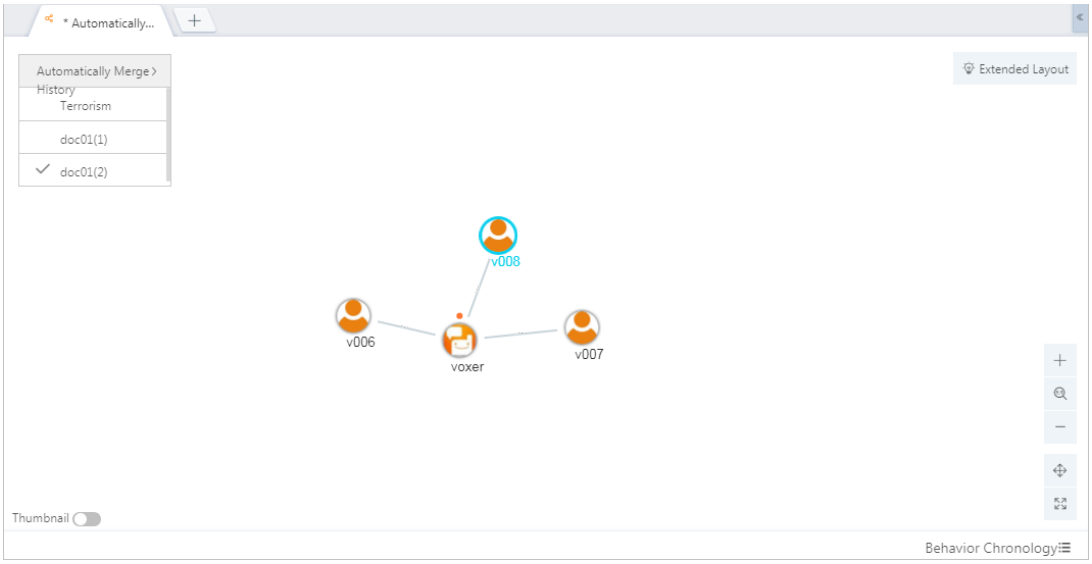
- Location: below the automatically merged file or the manually merged files.
- Name format: username + (version number). For example, test123(1).
- Author: the user who published the file.
- Description: displayed below the file name.
- Time: the time when the file was generated is displayed on the left side of the file.

8.13.3.5. Automatically merge files

When a new version is published, both the shared members and the sharer can use the auto merge feature to merge the new version with the earliest version (the initial file).

After a new version is published, the automatically merged file is updated, as shown in Automatically merge files.

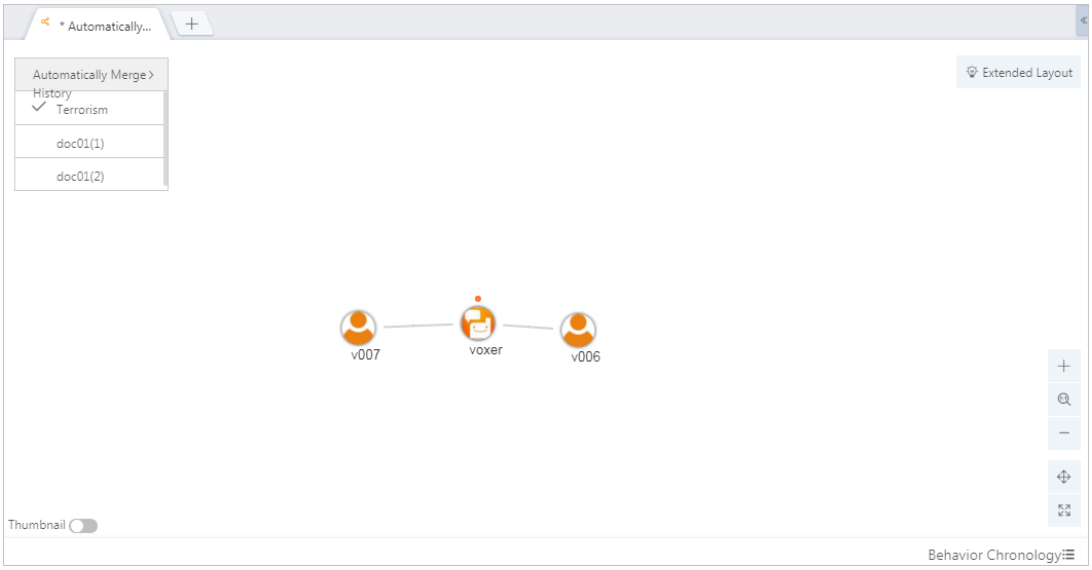
Automatically merge files



The versions are listed in the upper-left corner of the graph area. By default, the file of the latest version is displayed.

You can select an earlier version. Select a version in the **Auto Merge History** list to view the file of an earlier version, as shown in Select an earlier version.

Select an earlier version



8.13.4. View and manage received shared files

You can view and modify the analysis files shared by other users on the Shared with Me page. However, you cannot delete these files.

Prerequisites

You have received a shared file.

Context

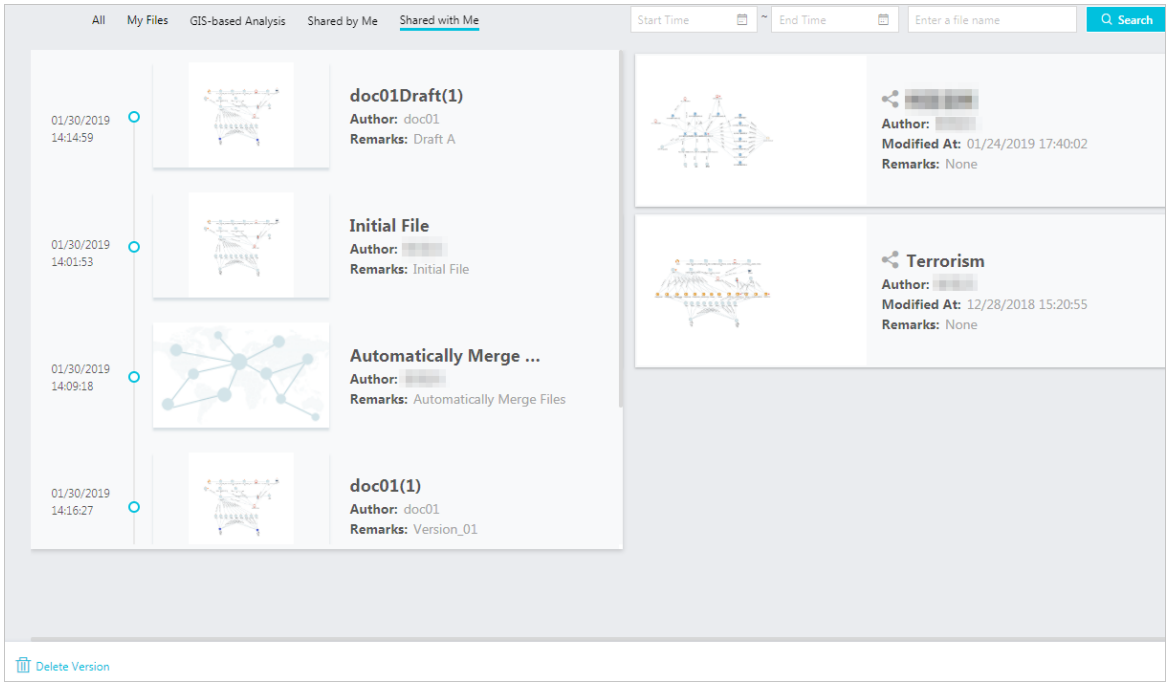
You can edit shared files, delete drafts, and publish new versions on the **Shared with Me** page.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **File Center**, and then click the **Shared with Me** tab. The **Shared with Me** page appears.

   The **Shared with Me** page displays all shared folders received by the current user. Each folder is a shared item. It contains multiple draft files, one initial file, one automatically merged file, multiple manually merged files, and multiple published version files.

   When a large number of folders or files are displayed, you can use the search function to query the target.



3. On the **Shared with Me** page, you can perform the following operations.

Operation	Description
Edit a shared file	Double-click an initial file, an automatically merged file, or a published version file to open the file in the graph area. Edit and save the file, and a draft file is generated.
Delete a draft	Select the draft to be deleted, click the **Delete Version** icon in the lower-left corner, and click **Delete** in the dialog box that appears.
Publish a version	See Publish a version.

8.14. Intelligent Network

8.14.1. Intelligent Network overview

This topic describes related concepts of Intelligent Network and how to use Intelligent Network.

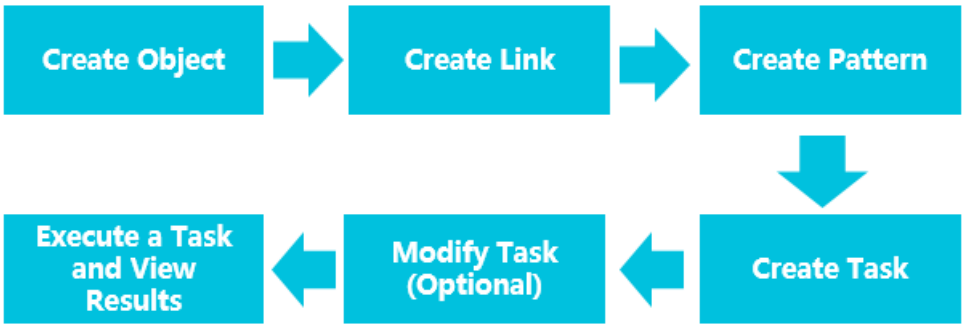
Concepts

Intelligent Network allows you to query subgraphs with the same graph structure as a task specified in a predefined pattern.

Intelligent Network involves the following concepts:

- **Pattern**: the relationship graph structure model that is predefined in Intelligent Network. Patterns are divided into private patterns and public patterns.
  - Private pattern: Only administrators and creators can use private patterns to create private tasks. Private patterns can be set to public patterns, but this is an irreversible operation.
  - Public pattern: All users can use public patterns to create public or private tasks. Public patterns cannot be set to private patterns.
- **Task**: created based on the pattern and used to query data with the same graph structure as the task in the data source. After a task is created based on a pattern, the task has the same settings as the pattern. You can modify the graph structure and filter conditions of the task. Tasks are divided into private tasks and public tasks.
  - Private task: Only administrators and creators can use private tasks. Private tasks created based on public patterns can be set to public tasks, but this is an irreversible operation.
  - Public tasks: All users can use public tasks. Public tasks cannot be set to private tasks.

Procedure to use Intelligent Network



8.14.2. Patterns

8.14.2.1. Create patterns

A pattern is the relationship graph structure model predefined in Intelligent Network. It is the basis of creating a task. Before you use the intelligent network to query a sub-graph, you must use related objects and links to predefine a pattern.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created an object and a first-degree link correlated with the object. For more information about how to create an object and a first-degree link, see Create an object and Create a first-degree link.

Context

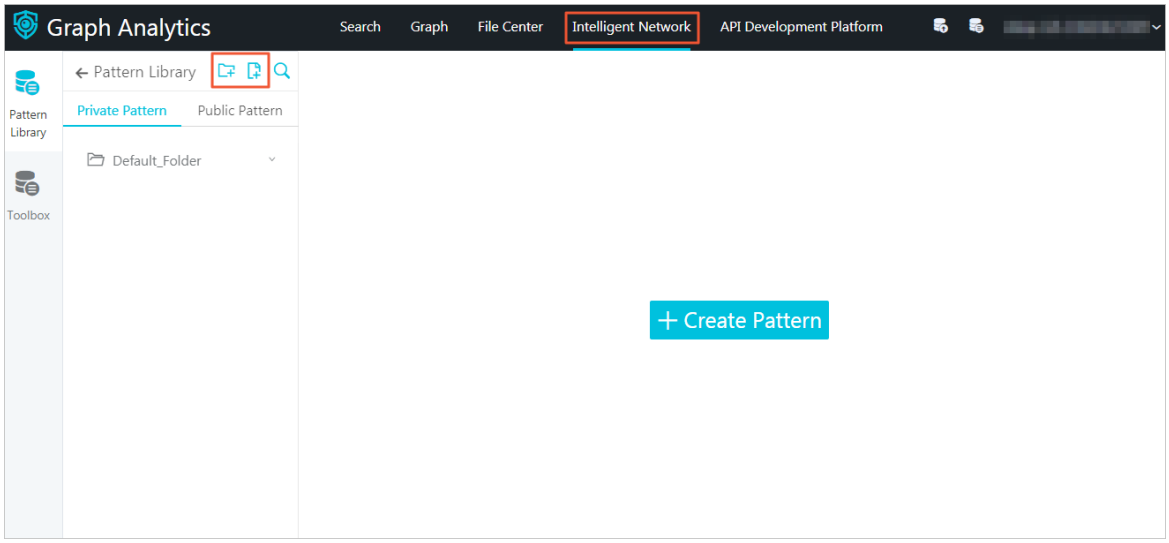
You can set the filter conditions in the **Property** tab and the **Global Conditions** tab based on your requirements.

- **Property** displays the properties of objects and links that can be used as query conditions. The **Conditional Query** properties and properties in **Accumulative Statistics Settings** will be displayed on the Property tab page.
- **Global Conditions** are based on the number and time type properties in a link. Global conditions take effect on the entire pattern. To configure **Global Conditions**, you must name the relevant number or time type properties on the **Property** tab.

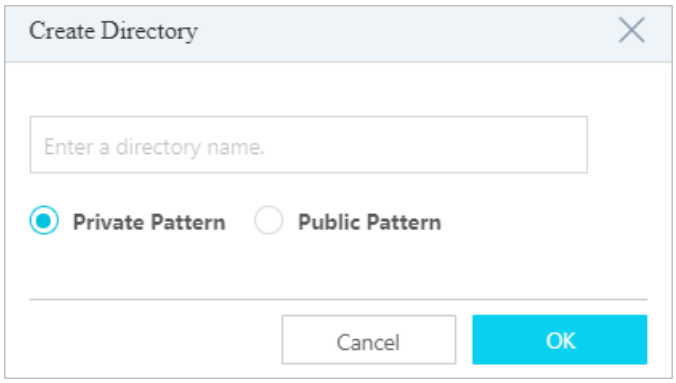
The following example shows a transfer pattern (A>B>C>A). The global condition is that the transfer from A to B is made earlier than the transfer from B to C.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.



3. You can create a new folder to save the new pattern separately.
   i. In the **Pattern Library** navigation pane, click the icon (**Create Folder**).
   ii. Specify the **Directory Name** and the pattern type as needed.



   iii. Click **OK**.
4. In the **Pattern Library** navigation pane, click the icon (**Create Pattern**) or **Create Pattern** in

the blank area on the right side.

5. Configure the relationship graph structure model of a pattern.

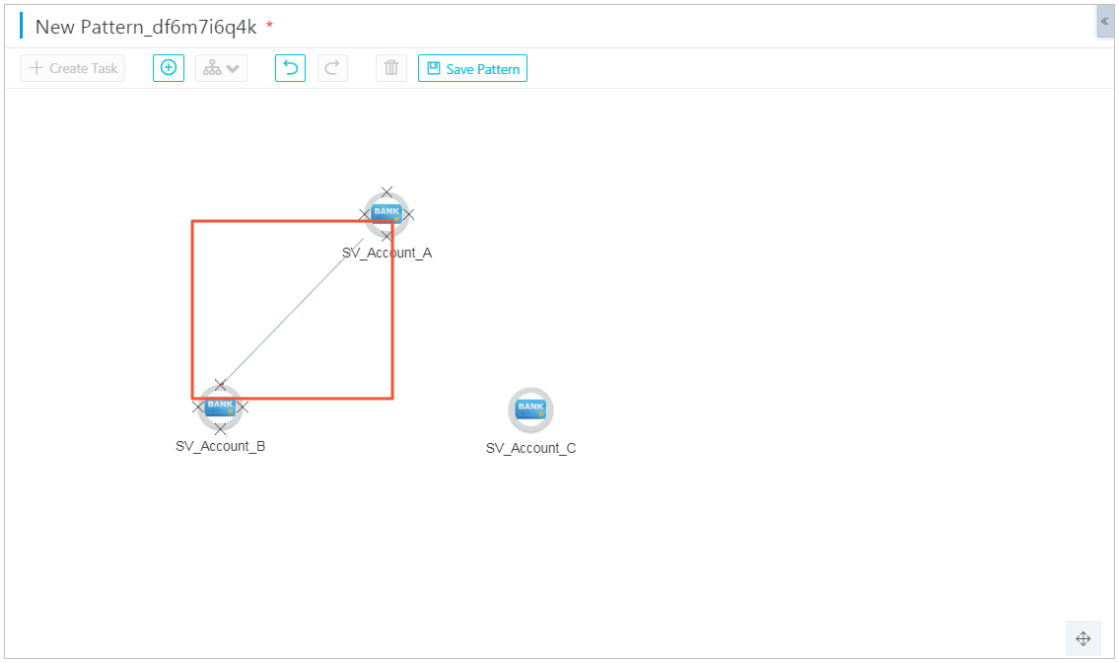
   If an incorrect operation occurs, you can click the icon (Undo) and the icon (Redo) to quickly restore to a specific state.

   i. For the new pattern **New pattern_xxx**, click the icon in the toolbar, or click **Toolbox** in the left-side navigation pane to open the tool box.

   ii. Drag the objects one by one from the toolbox to the right side of the page to add object nodes to **New mode_xxx**.

   iii. Right-click the object node and select **Change Node Name** to set the name of the object node.

   iv. Move your mouse pointer to the source object. When four cross signs appear around the object, move your mouse pointer to a cross sign and press the left mouse button. Then, drag the mouse pointer to the target object. When four cross signs appear around the target object, release the left mouse button.



   v. In the **Add Link** dialog box that appears, select the link type and specify the link logic.

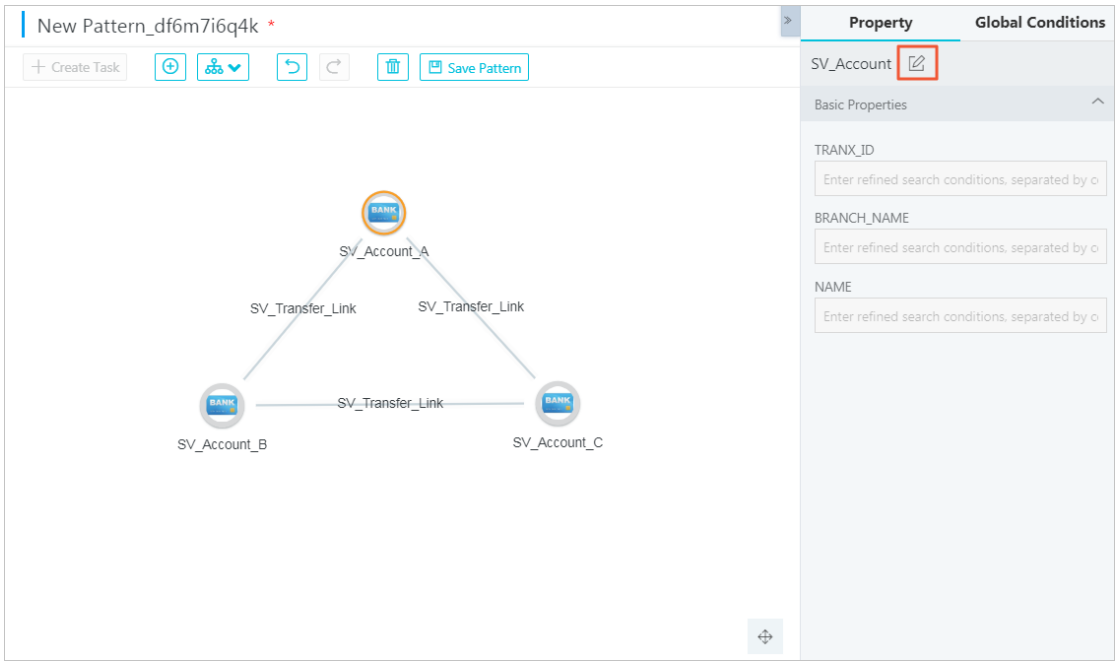
   vi. Click **OK**.

   vii. (Optional)To handle multiple object nodes with complex relationships, you can select the corresponding object node and click the icon to select an appropriate layout.

6. Configure the filter conditions in the **Property** tab.

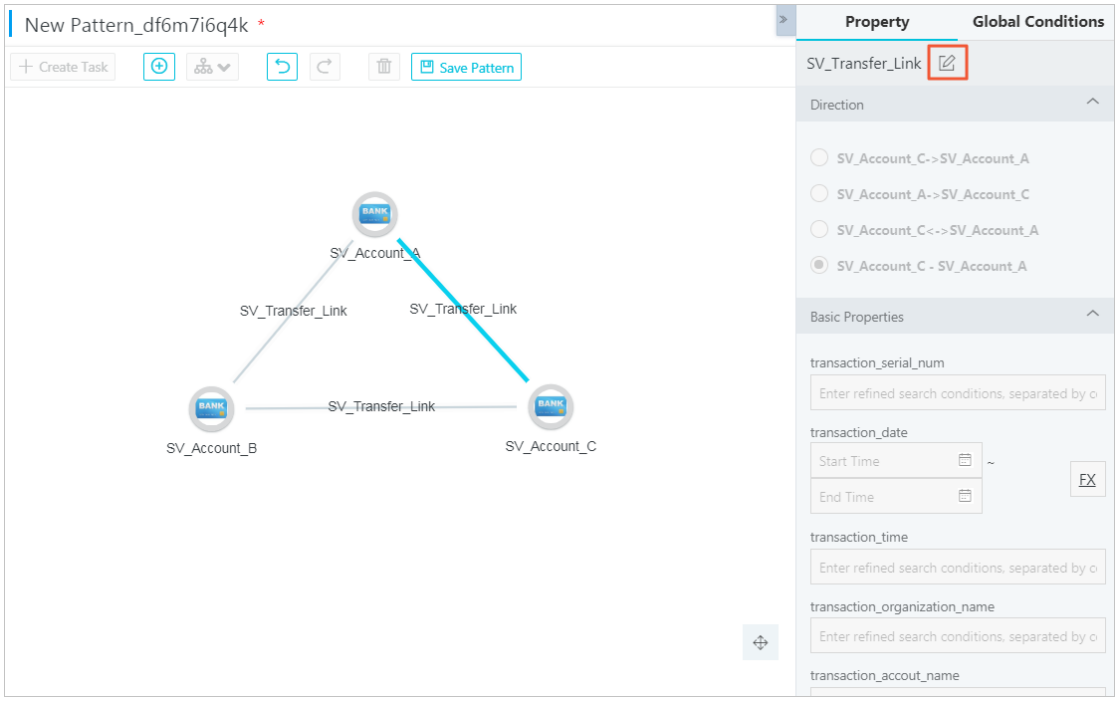
   i. Click the icon in the upper-right corner, and then click **Property**.

ii. Select an object. On the **Property** tab, click the icon to set the filter conditions for the object.



iii. After you have configured the preceding parameters, click the icon to save the **Property** configurations of the object.

iv. Select a link. On the **Property** tab, click the icon to set the filter conditions and direction of the link.



On the **Property** page of a link, properties marked with **FX** are the basis of **Global Conditions**.

v. To set the global conditions based on a property marked with **FX**, click **FX** to name the property.

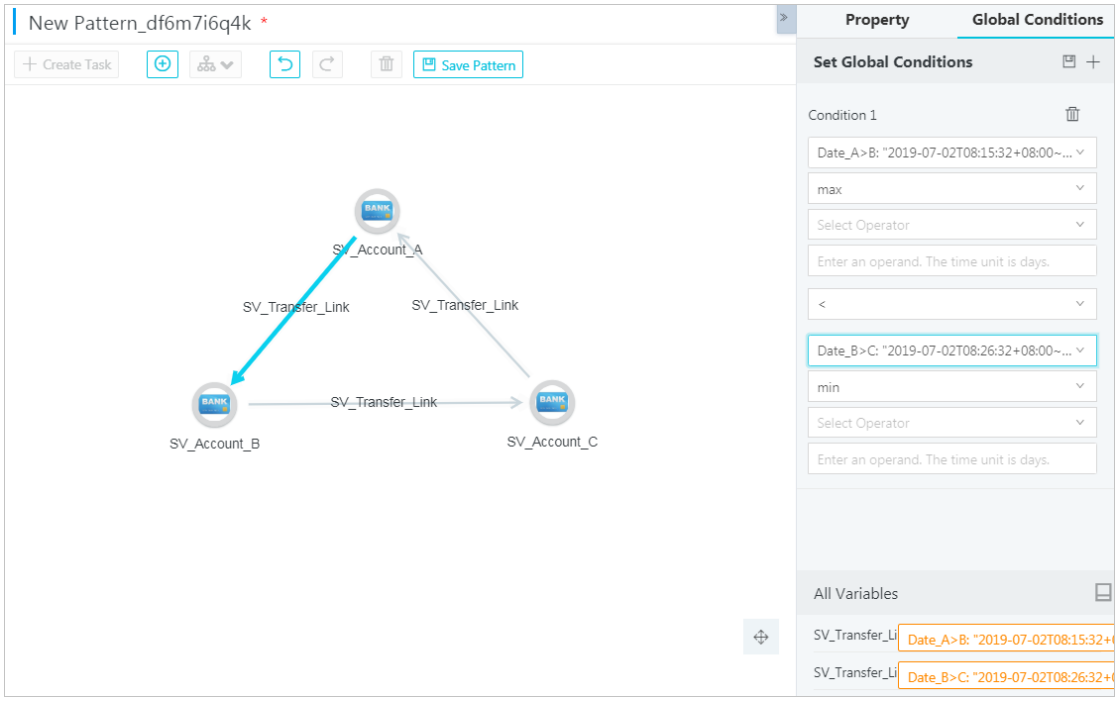
In this example: the **Transaction Date** property of the link between A and B is named as **Date A>B**. The **Transaction Date** property of the link between B and C is named as **Date B>C**.

vi. After you have configured the preceding parameters, click the icon to save the **Property** configurations of the link.

7. Configure filter conditions on the **Global Conditions** tab.

i. Click the icon in the upper-right corner, and then click **Global Conditions**.

ii. If you have not set the global conditions, click **Click here to configure settings**.

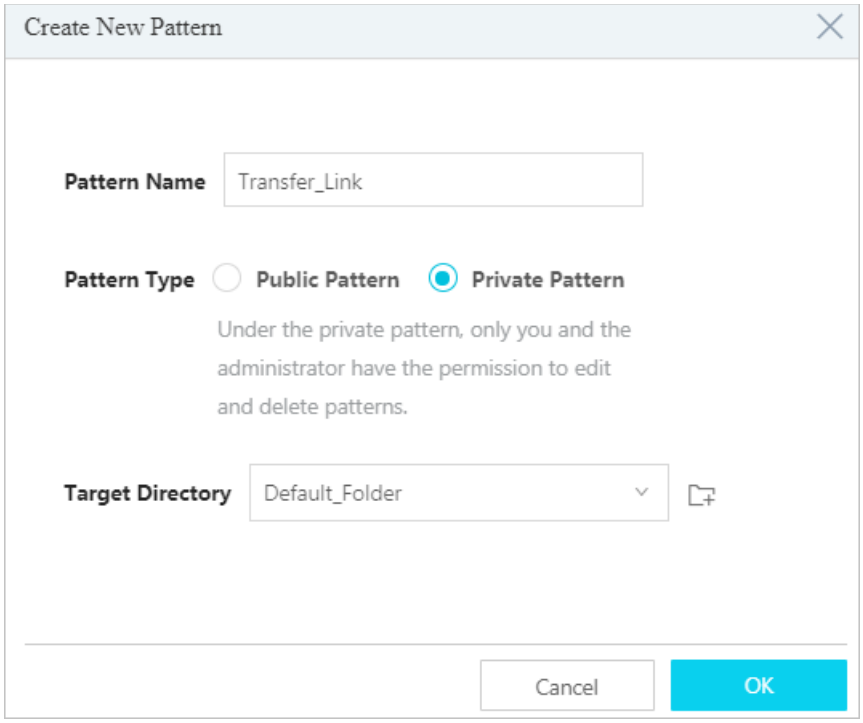


On the **Global Conditions** tab, configure the global conditions based on the link properties that are marked with **FX** in the **Property** tab. The global condition in this example is that the transfer from A to B is made earlier than the transfer from B to C.

iii. After you have configured the preceding parameters, click the icon to save the configurations on the **Global Conditions** tab.

8. Click **Save Pattern**. Set the parameters in the **Create Pattern** dialog box that appears.

If you cannot find a proper folder in **Target Directory**, you can click the icon to create a folder.



9. Click **OK**.

8.14.2.2. View patterns

In Graph Analytics, you can view all the patterns within your permissions and details of each pattern, such as the relationship graph structures and tasks created based on the pattern.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can view the private patterns.

- You have created a pattern. For more information about how to create a pattern, see Create patterns.

Context

You can view the following pattern information:

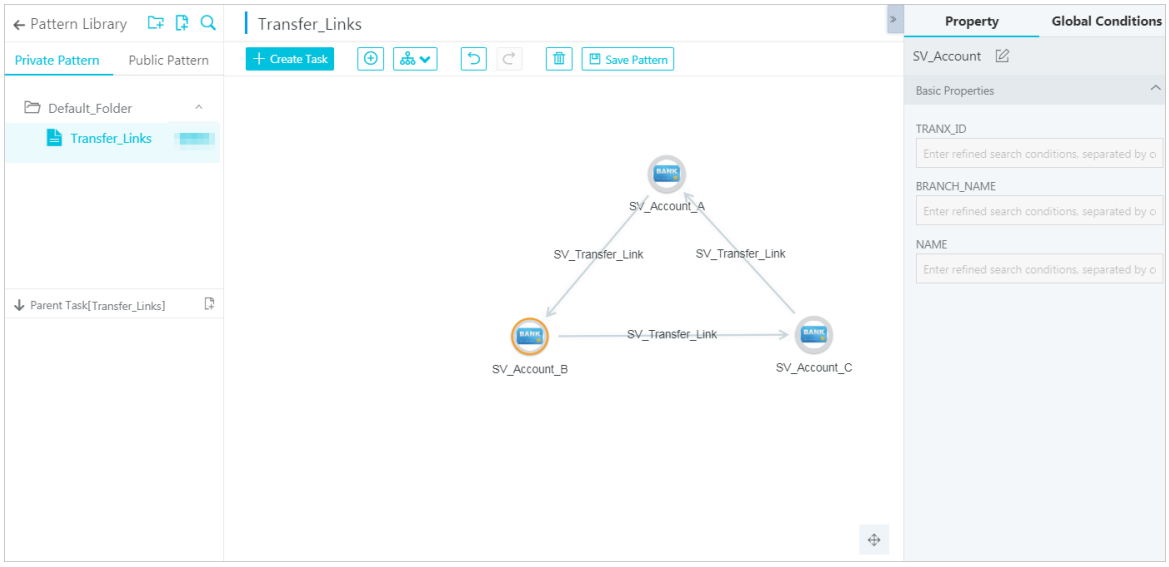
- Relationship graph structures of a pattern
- Tasks created based on a pattern
- Filter conditions of a pattern

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, click **Private Pattern**, and then click a folder to view the private patterns in the folder.Click **Public Pattern**, and then click a folder to view the

private patterns in the folder.

4. Click a pattern to view its details.



Operation	Procedure
View the relationship graph structures	Click a pattern to view the relationship graph structures on the right side of the page.
View tasks created based on a pattern	Click a pattern. All tasks created based on this pattern are displayed under the **Pattern Library** tab.
View the filter conditions of a pattern	In the relationship graph structure, select an object or link, and click the icon in the upper-right corner to display the **Property** tab and the **Global Conditions** tab. You can view the filter conditions of objects or links and the global filter conditions of the pattern.

8.14.2.3. Modify patterns

You can modify a pattern when its relationship graph structure and filter conditions are not suitable for the business scenarios. Modifying a pattern does not affect the tasks that are created based on the pattern.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can modify private patterns.

- You have created a pattern. For more information about how to create a pattern, see Create patterns.

Context

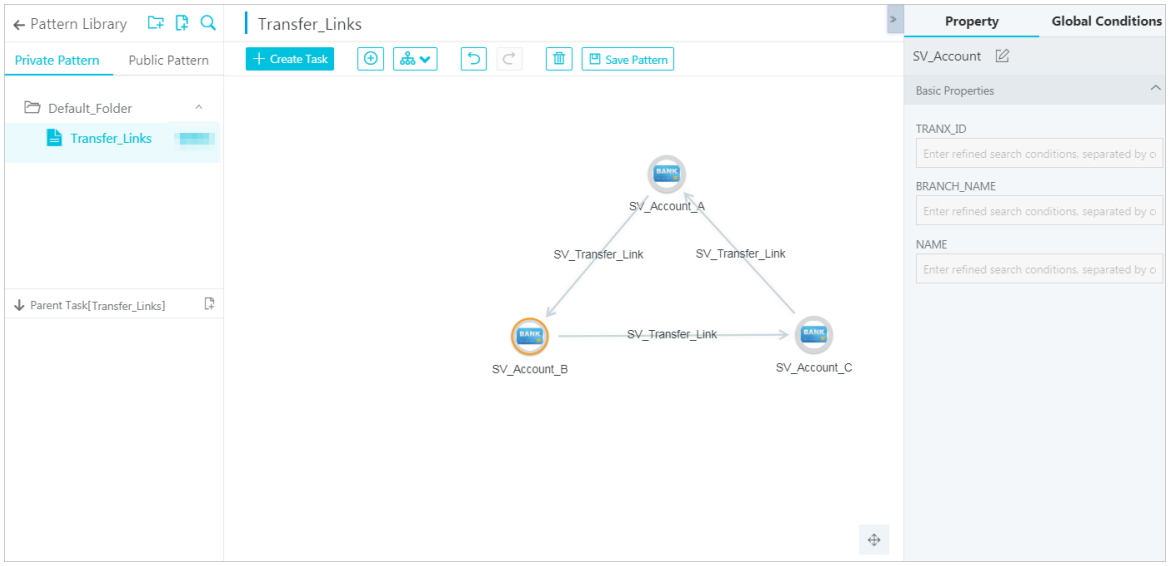
In Graph Analytics, you can modify the following aspects of a pattern:

- Modify the pattern name
- Modify the filter conditions
- Modify the object node name
- Modify the link type
- Add an object node
- Add a link
- Delete an object node
- Delete links

> **Note** After you modify a pattern, the existing tasks that are created based on this pattern are not affected. The modifications will be applied when you create a new task based on the modified pattern.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern**.



4. Modify the pattern as needed.

Operation	Procedure
Modify the pattern name	i. In the **Pattern Library** navigation pane, click **Private Pattern** or **Public Pattern** and select a pattern to be modified. Right-click the pattern and select **Rename**. ii. Reset the name, and then press **Enter**. A message is displayed, indicating that the operation is successful.

Operation	Procedure
Modify the filter conditions	Click the icon in the upper-right corner, and set the filter conditions in **Property** and **Global Conditions**. For more information, see Create patterns.
Modify the object node name	i. Right-click the object node to be modified, and select **Change Node Name**. ii. In the dialog box that appears, set the new name of the object node, and then click **OK**.
Modify the link type	i. Right-click the link to be modified, and select **Set Link Type**. ii. In the dialog box that appears, select the link type, specify the link logic, and then click **OK**. You can select multiple link types.
Add an object node	i. Click the icon in the toolbar, or click **Toolbox** on the left side of the page. ii. Drag the objects one by one from the toolbox to the right side of the page to add object nodes. iii. Add a link to a new object node.
Add a link	After you have deleted a link or added a new object node, you need to add a link for the object node. i. Move your mouse pointer to the source object. When four cross signs appear around the object, move your mouse pointer to a cross sign and press the left mouse button. Then, drag the mouse pointer to the target object. When the four cross signs appear around the target object, release the left mouse button. ii. In the **Add Link** dialog box that appears, select the link type and specify the link logic. iii. Click **OK**.

Operation	Procedure
Delete object nodes or links	Note After an object node is deleted, the links related to the object node are also deleted. Method one: i. Select an object node or link, and click the icon in the toolbar. Or, you can press the **Delete** key. You can click-and-drag the mouse to box select multiple object nodes. ii. In the message box that appears, click **Delete Selected**. Method two: Right-click an object node or link, or any of the object nodes in the box selection, and then select **Delete Selected** to directly delete the object node or link.

5. After you have configured the preceding parameters, click **Save Pattern** in the toolbar to save the pattern.

8.14.2.4. Set private patterns to public patterns

In Graph Analytics, you can set private patterns to public patterns, but this is an irreversible operation.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  Note Only administrators and creators can set private patterns to public patterns.

- You have created a private pattern. For more information about how to create a pattern, see Create patterns.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, click **Private Pattern**, right-click a private pattern, and then select **Set as Public Pattern**.
4. In the dialog box that appears, configure relevant parameters.

   The parameters are described in Parameter configurations for setting a private pattern to a public pattern.

Parameter configurations for setting a private pattern to a public pattern

Parameter	Description
Publish the dependent task?	Valid values are as follows: ○ **Yes**: Private tasks created based on this private pattern are also set to public tasks. ○ **No**: Private tasks created based on this private pattern remain private tasks. After this private pattern is set to a public pattern, these tasks will be converted to private tasks that are created based on a public pattern. You can manually set these tasks as public tasks when necessary.
Target Directory	Set the target directory in **Public Pattern** to receive the private pattern. When the private pattern is set as a public pattern, it will be moved from the directory in **Private Pattern** to the specified directory in **Public Pattern**.

5. After you have configured the preceding parameters, click **OK**. A message is displayed, indicating that the operation is successful.

8.14.2.5. Delete a pattern

You can delete a pattern that is no longer used. Deleting a pattern does not affect the tasks that are created based on the pattern.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can delete private patterns.

- You have created a pattern. For more information about how to create a pattern, see Create patterns.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, click **Private Pattern** or **Public Pattern** and select a pattern to be deleted. Right-click the pattern and select **Delete**.
4. In the message box that appears, click **OK**.

8.14.3. Tasks

8.14.3.1. Create a task

A task is created based on a pattern. It can be used to query the data with the same graph structure as the task in the data source.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.
- You have created a pattern. For more information about how to create a pattern, see Create patterns.

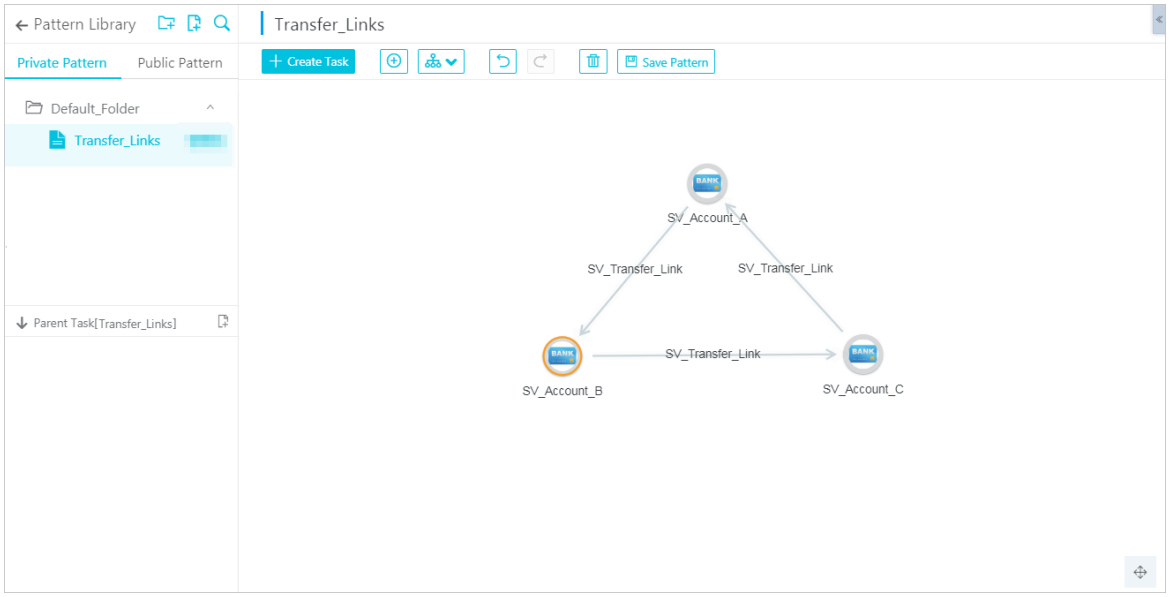
Context

For a task created based on the pattern, the objects, links, and filter conditions of the task are inherited from the pattern by default, which are exactly the same as the pattern. To modify these configurations, see Modify a task.

Procedure

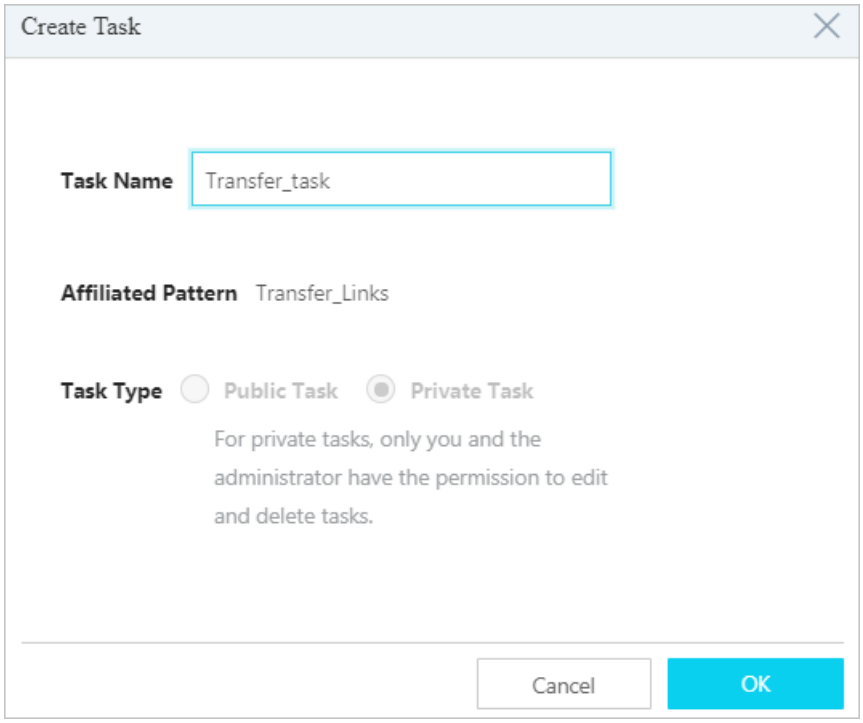
1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern**. Click **Create Task** on the right side of the page or click the icon (**Create a Task**).

   > **Note** You can only create private tasks based on a private pattern. You can create public tasks or private tasks based on a public pattern.



4. Configure the parameters on the **Create Task** dialog box that appears.

   The parameters are described in Parameter configurations for creating a task.



Parameter configurations for creating a task

Parameter	Description
Task Name	The user-defined task name.
Affiliated Pattern	The pattern that you have selected. It is automatically entered and cannot be modified.
Task Type	When the selected pattern is public, you can set the task type to **Public Task** or **Private Task**. When the selected pattern is private, the default task type is **Private Task** and cannot be modified.

5. Click **OK**.

   After you have configured the preceding parameters, a message appears, indicating that the task has been created. The objects, links, and filter conditions of the task are inherited from the pattern by default, which are exactly the same as the pattern.

8.14.3.2. Check the task

You can view tasks based on patterns. You can select a specific pattern and view all the tasks that are created based on the pattern. You can also view the detailed information about these tasks, such as the relationship graph structures and filter conditions of the tasks.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can view the private tasks.

- You have created a task. For more information about how to create a task, see Create a task.

Context

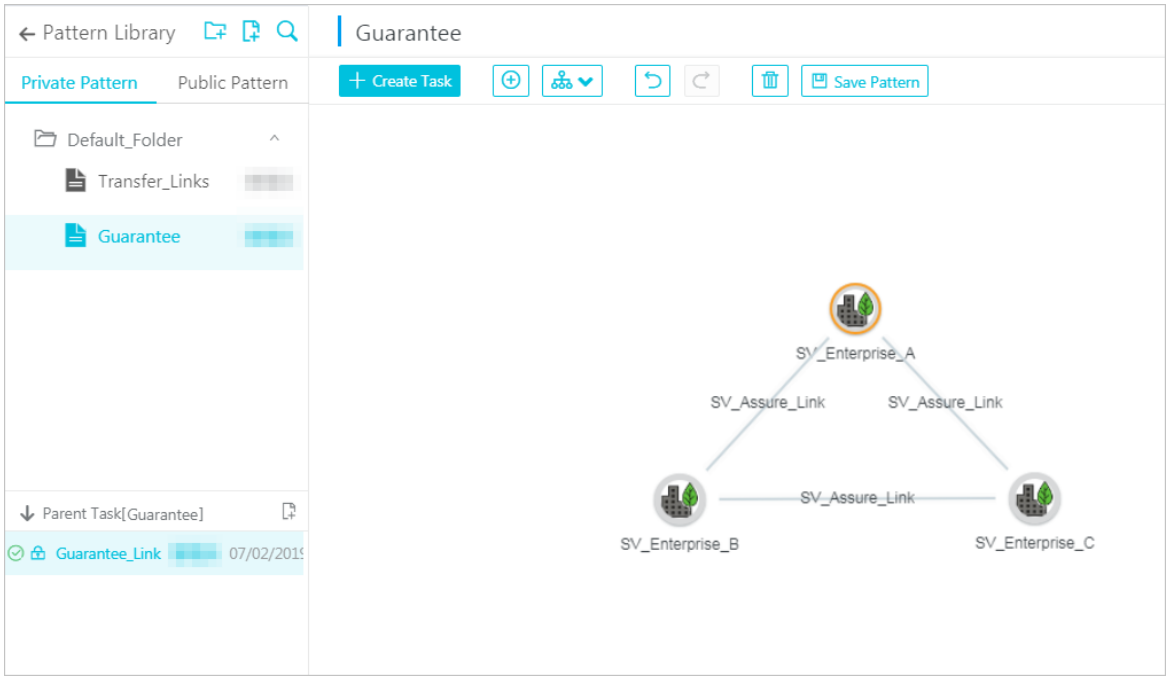
Tasks are divided into public tasks and private tasks. A lock icon is displayed before a private task, but no lock icon is displayed before a public task. A task can have three statuses: executed, not executed, and failed. These three statuses are represented by different icons.

You can view the following task information:

- The task type and task status
- Relationship graph structures of a task
- Filter conditions of a task
- Primary keys of the object in the task

Procedure

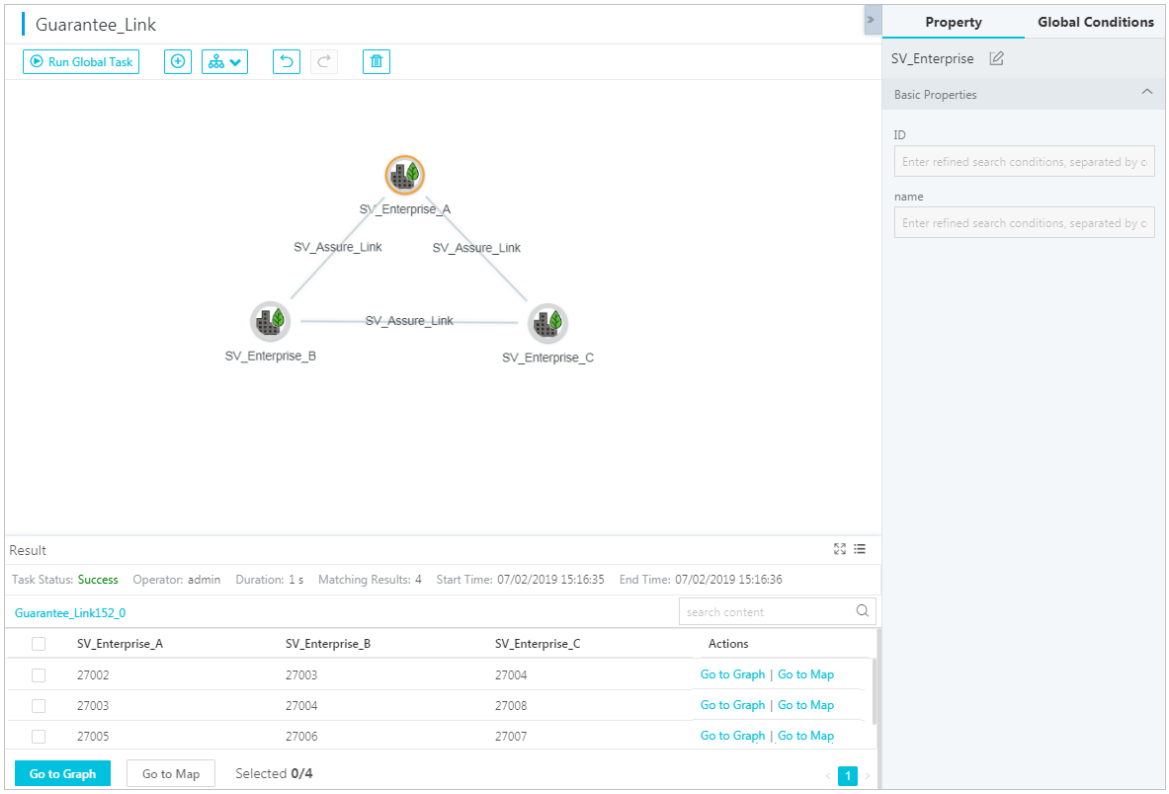
1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern** and view the tasks that are created based on the pattern.



The task types and statuses are described as follows:

- Tasks are divided into public tasks and private tasks. A lock icon is displayed before a private task, but no lock icon is displayed before a public task.
- A task can have three statuses: executed, not executed, and failed. These three statuses are represented by different icons.

4. Click a task and view the detailed information about the task as follows.



Operation	Procedure
View the relationship graph structures of a task	Click a specific task to view the relationship graph structures on the right side of the page.
View the filter conditions of a task	Select an object or a link, and click the icon in the upper-right corner. The **Property** and **Global Conditions** tabs are displayed. You can view the filter conditions of an object or a link and the global conditions of tasks.
View the primary keys of objects in a task	Right-click an object and select **Set Primary Keys** to view the primary keys that have been set for this object.

8.14.3.3. Modify a task

For a task that is created based on a pattern, the objects, links, and filtering conditions of the task are inherited from the pattern by default, which are exactly the same as the pattern. When some configurations of a task do not match the scenario, you can modify the task accordingly.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

> **Note** Only administrators and creators can modify private tasks.

- You have created a task. For more information about how to create a task, see Create a task.

Context

In Graph Analytics, you can modify the following aspects of a task:

- Modify the filter conditions
- Modify the object node name
- Modify the link type
- Add an object node
- Add a link
- Delete an object node
- Delete a link

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern**. Select the task to be modified.
4. Modify a task as shown in the following operations.

Operation	Procedure
Modify the filter conditions	You can click the icon in the upper-right corner. You can set the filter conditions for a task in the **Property** and **Global Conditions** tabs that appear. For more information, see Create patterns.
Set the object primary key	i. Right-click the object node to be modified, and select **Set Primary Keys**. ii. In the dialog box that appears, set the primary keys of the object node, and click **OK**. You can set multiple primary keys.
Modify the object node name	i. Right-click the object node to be modified, and select **Change Node Name**. ii. In the dialog box that appears, set the new name of the object node, and then click **OK**.
Modify the link type	i. Right-click the link to be modified, and select **Set Link Type**. ii. In the dialog box that appears, select the link type, specify the link logic, and then click **OK**. You can select multiple link types.

Operation	Procedure
Add an object node	i. Click the icon in the toolbar, or click **Toolbox** on the left side of the page. ii. Drag the objects one by one from the toolbox to the right side of the page to add object nodes. iii. Add a link to a new object node.
Add a link	After you have deleted a link or added a new object node, you need to add a link for the object node. i. Move your mouse pointer to the source object. When four cross signs appear around the object, move your mouse pointer to a cross sign and press the left mouse button. Then, drag the mouse pointer to the target object. When the four cross signs appear around the target object, release the left mouse button. ii. In the **Add Link** dialog box that appears, select the link type and specify the link logic. iii. Click **OK**.
Delete object nodes or links	Note After an object node is deleted, the links related to the object node are also deleted. Method one: i. Select an object node or link, and click the icon in the toolbar. Or, you can press the **Delete** key. You can click-and-drag the mouse to box select multiple object nodes. ii. In the message box that appears, click **Delete Selected**. Method two: Right-click an object node or link, or any of the object nodes in the box selection, and then select **Delete Selected** to directly delete the object node or link.

5. Click **Run Global Task** to save the modification configuration.

Note After a task is modified, you must execute the task again. Otherwise, the modifications cannot be saved.

8.14.3.4. Execute the task and view the result

After the task is created or modified, you can execute the task to query sub-graphs that have the same graph structure as the task. The task execution results are displayed in a list. You can also view the execution results in the graph area.

Prerequisites

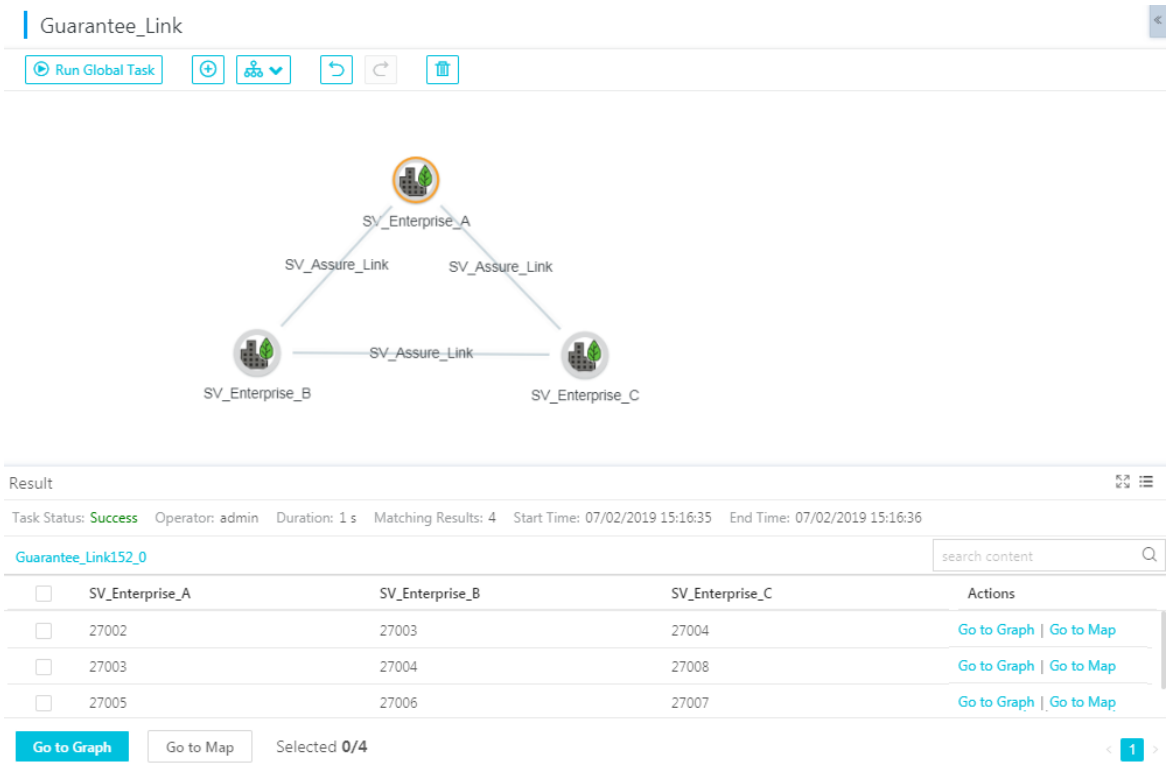
- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can execute private patterns.

- You have created a task. For more information about how to create or modify a task, see Create a task and Modify a task.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern**. Select the task to be executed.
4. Click **Run Global Task** in the top toolbar.
5. In the right-side area, click the icon to view the execution results.

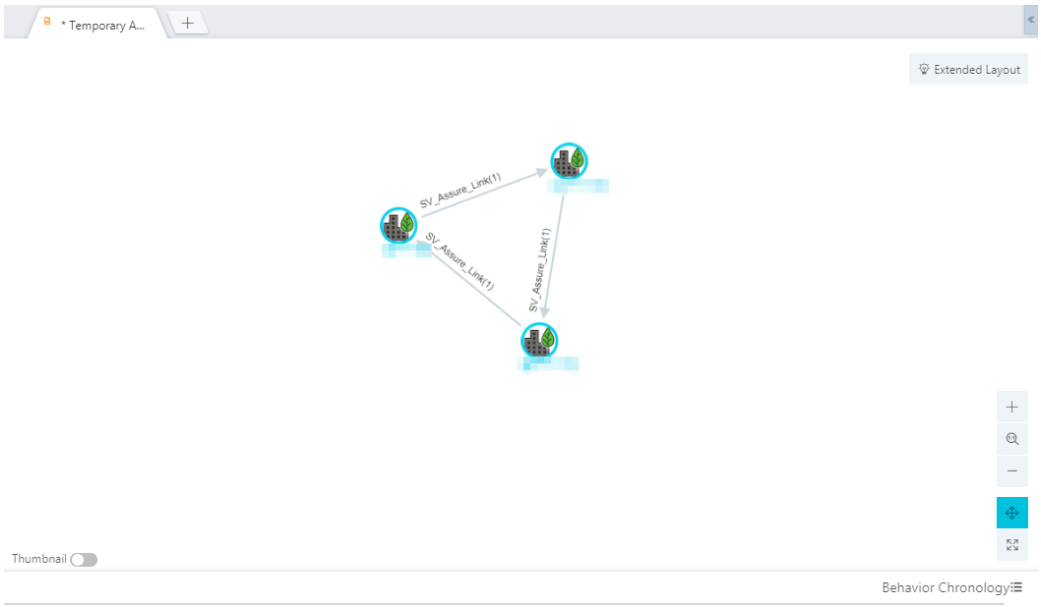


6. View the task execution results in the graph area.

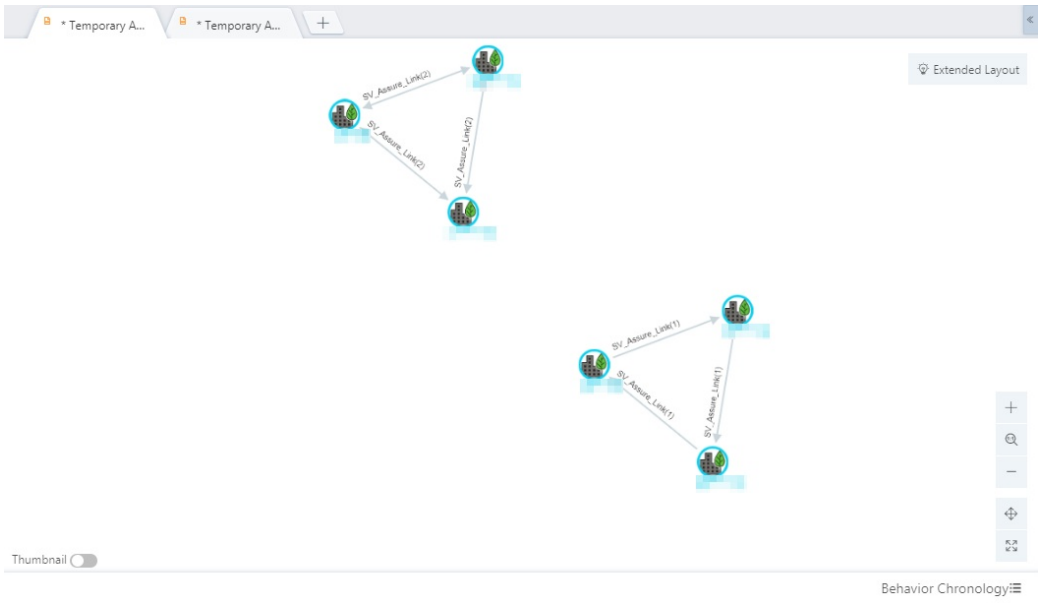
   > **Note** The task result is displayed as a temporary analysis file in the graph area. You can save and share the file and perform other operations. You can also extend the analysis.

Operation	Procedure
View the results of a single task in the graph area	Select a record and click **Go to Graph**. The result is displayed on the graph page, as shown in View the results of a single task in the graph area.
View the results of multiple tasks in the graph area	You can also select multiple records and then click **Go to Graph**.The results are displayed on the graph page, as shown in View the results of multiple tasks in the graph area.

View the results of a single task in the graph area



View the results of multiple tasks in the graph area



8.14.3.5. Set a private task as a public task

Private tasks created based on public patterns can be set to public tasks, but this is an irreversible operation.

Prerequisites

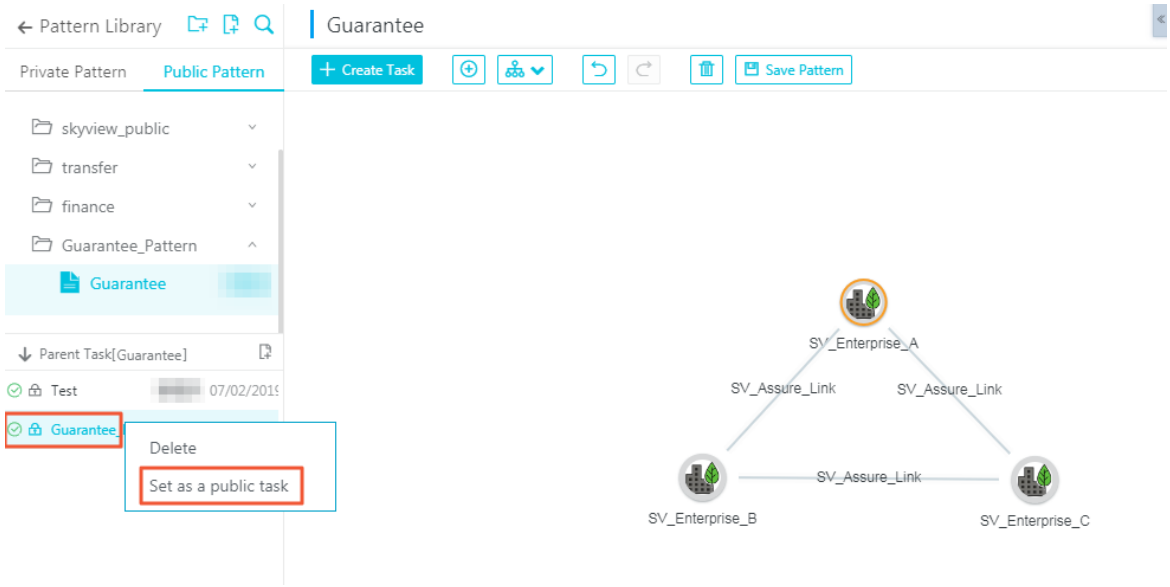
- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can set private patterns to public patterns.

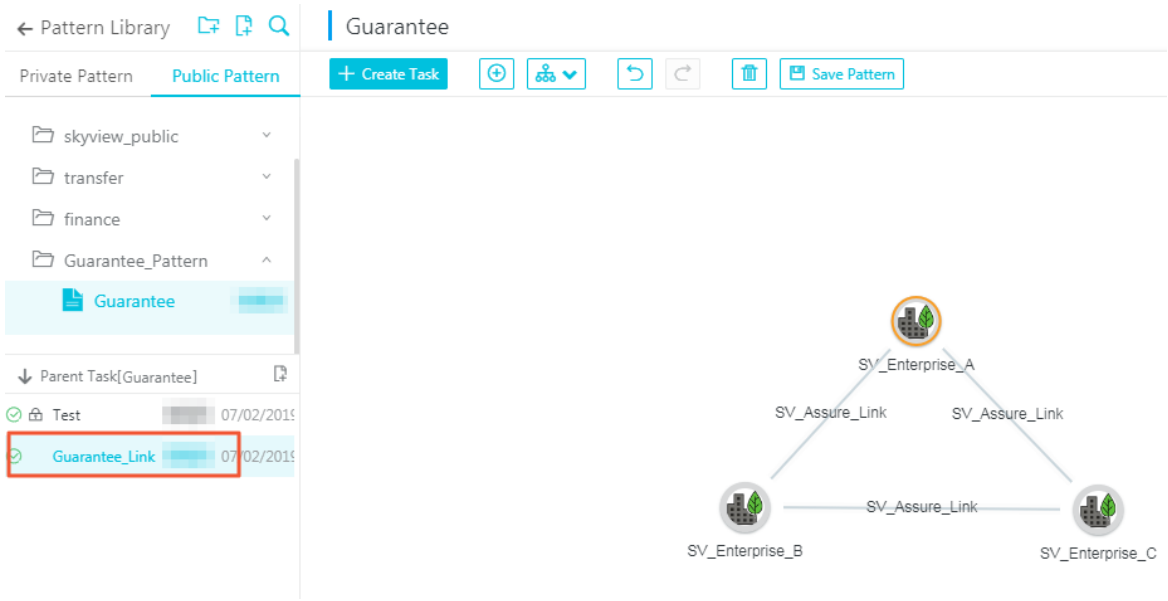
- You have created a private task based on the public pattern. For more information about how to create a pattern, see Create a task.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, click **Public Pattern** . Right-click a private task that is created based on the pattern, and then select **Set as Public Task**.



After a private task is set to a public task, the lock icon in front of the task disappears. A message is displayed, indicating that the operation is successful.



8.14.3.6. Delete a task

You can delete a task when it is no longer needed.

Prerequisites

- Make sure that you have an account and password for Graph Analytics and are authorized to operate on Analytics Workbench and Administration Console.

  > **Note** Only administrators and creators can delete private patterns.

- You have created a task. For more information about how to create a pattern, see Create a task.

Procedure

1. Log on to Analytics Workbench.
2. In the top navigation bar, click **Intelligent Network**.
3. In the **Pattern Library** navigation pane, select a pattern in **Private Pattern** or **Public Pattern**. Right-click the task to be deleted and then click **Delete**.

   > **Note** Deleted tasks cannot be restored. Exercise caution when you delete a task.

4. In the dialog box that appears, click **OK**.

8.15. Examples

8.15.1. Tax industry case studies

Tax inspectors used Graph Analytics to analyze abnormal information and uncovered a tax cheat group that was related to 13 companies.

Case background

On April 1, 2016, a clerk of an overseas trading platform was routinely checking goods being loaded and unloaded. When the clerk checked “Hangzhou Children's Products Co., Ltd.”, he was told that the goods had been shipped. The company exported goods such as children's products to overseas countries with a total volume of 40 million RMB. Apparently, this was a case of avoiding "loading inspection", and this case attracted the clerk's attention. The clerk used Graph Analytics to analyze the company and found a large-scale tax cheat group related to this company.

Data preparation

> **Note** All the data in this case, including individual people's names, company names, places, and times, is purely fictitious.

Data resources used in this case include:

- Information of the drawer (the manufacturer), the customer (the domestic trader), the remitter, and the overseas trader.
- The relationships between the customer and the drawer, the customer and the remitter, and the customer and the overseas trader are required.

Connect to data sources and configure an OLEP model

Before you perform an analysis, you must connect the data source to Graph Analytics and build an OLEP model based on the data table in the data source.

1. Log on to Administration Console of Graph Analytics. For more information about how to connect the data source to Graph Analytics, see Create data sources.
2. Create objects based on the data table in this case, as shown in Create OLEP models for tables.

   The created objects are as follows.

Objects mapped by the data table

Data table	Object
Drawer information table	○ Drawer: Maps the drawer information. ○ WIFI: Maps the WIFI used by the drawer.
Customer information table	○ Contact phone number: Maps the telephone number of the customer. ○ Customer: Maps the customer information.
Remitter information table	Remitter: Maps the remitter information.
Overseas trader information table	○ Country: Maps the country where the overseas trader is located. ○ Overseas trader: Maps the overseas trader information.

3. Create a first-degree link mapping for the data table in this case, as shown in Create OLEP models for tables.

The first-degree link is created as follows.

Links mapped by the data table

Data table	First-degree link	Source object	Target object
Drawer information table	○ Frequent logon WIFI: The link between the drawer and the frequent logon WIFI can be used to query both the drawer and the WIFI information. ○ Name of the drawer's company: The link between the drawer and the drawer's company can be used to query both the drawer and the company name. ○ Drawer's phone number: The link between the drawer and the drawer's phone number can be used to query both the drawer and the phone number of the drawer.	○ Frequent logon WIFI: The drawer ○ Name of the drawer's company: The drawer	○ Frequent logon WIFI: The WIFI ○ Name of the drawer's company: The name of the company
Customer information table	○ Customer's phone number: The link between the customer and the customer's phone number can be used to query both the telephone number or the user who have used this phone number. ○ Name of the customer's company: The link between the customer and the customer's company can be used to query both the customer or the company of the customer.	○ Customer's phone number: The customer ○ Name of the customer's company: The customer	○ Customer's phone number: The phone number ○ Name of the customer's company: The name of the company
Customer-drawer information table	Invoicing: This link can be used to query both the customer and the drawer.	Drawer	Customer

Data table	First-degree link	Source object	Target object
Customer-remitter information table	Remittance: This link can be used to query both the customer and the remitter.	Remitter	Customer
Customer-overseas trader information table	Purchase: This link can be used to query both the overseas trader and the customer.	Overseas trader	Customer
Overseas trader information table	Country: This link can be used to query both the overseas trader and the country where the overseas trader is located.	Overseas trader	Country

4. For more information about how to configure the business information related to these objects and links, see Configure object properties and business parameters and Configure link properties and business parameters.
5. Create multi-degree links as shown in Create a multi-degree link.

   The multi-degree links created in this case are as follows.

Multi-degree links

Multi-degree link	Base links	Description
The drawer and the customer with the same name.	○ Name of the customer's company. ○ Name of the drawer's company.	Queries the drawer with the same name as the customer.

Analysis process

After you have connected the data source and configured the OLEP model, you can go to Analytics Workbench to analyze cases.

1. Log on to Analytics Workbench.
2. On the **Graph** page, create a new analysis, and add **Hangzhou Children's Products Co., Ltd** as the node to start with.
3. Select the added node, and query the company's information on the right side of the page.

   Results: This company is a manufacturer company instead of a trading company.

   Inferences made by analysts: This company needs a domestic trader to export its goods.
4. Analysts used the link extension feature of Graph Analytics to query the downstream customer of this company, namely the domestic trader.

   Select this company as a node, and click **Link Extension** in the toolbar. Set **Link Type** to**Invoicing** to query the downstream customer of this company.

Results: This company has been the drawer for three trading companies. These three companies are from the medical industry and the apparel industry, but somehow have become the drawee of a manufacturer engaged in children's products.

5. Analysts used the link extension feature to investigate these three customers to find the overseas traders that have built business relationships with these three customers.

   Select these three customers, and click **Link Extension**. Set **Link Type** to **Purchase** to query the overseas traders that have built business relationships with these three customers.

   Results: All these customers have their own overseas trader.

6. Analysts used the link extension feature to search the remitter of these three customers.

   Select these three customers, and click **Link Extension**. Set **Link Type** to **Remitter** , and search for the remitters related to these three customers.

   Results: The analysis results indicate that these three customers receive remittance from the same remitter. In this phase, the relationship network is downward-trend and looks like a symmetrical funnel, which is a typical abnormal pattern.

   Inferences made by analysts: A manufacturer sells products to overseas buyers through three customers, but there is only one remitter. Whether these three overseas traders are in the same area or not, this network pattern is very suspicious.

7. Analysts tried to query the locations of these three overseas traders.

   Select these overseas traders, and click **Link Extension**. Set **Link Type** to **Country** to query the locations of these three overseas traders.

   Results: These three overseas traders are located in different countries.

   Inferences made by analysts: These overseas traders are located in different countries, but the remittance is made by the same remitter. Given that, it is very likely that these three customers and the drawer have committed tax cheating.

8. To obtain more information, analysts viewed the information cubes of these three customers.

   Results: Two of the customers are companies that provide both manufacturing and trading services.

9. Following these clues, analysts queried the manufacturers of these two companies.

   Select these two companies and click **Link Extension**. Set **Link Type** to **Drawer and customer with the same name** to query the manufacturers of these two companies.

   Results: These two companies share the same manufacturer (drawer).

10. Analysts used **Group Analysis** to analyze the relationship between these two companies.

    Select these two nodes and their manufacturers, and choose **Analyze** > **Group Analysis**. Set **Link Type** to **Invoicing** to analyze the relationship between them.

    Results: These two companies issue invoices to each other.

    Judgement made by the analysts: Based on experience, this phenomenon is obviously abnormal.

11. Analysts used **Backbone Analysis** to verify whether this network contains any key members.

    In the toolbar, choose **Analyze** > **Backbone Analysis**. Set the **Backbone Node** to **Customer** to analyze the backbone customers in the current network.

Results: The current relationship network contains two companies as key nodes. These two companies have taken up important positions in the current relationship network.

12. Analysts used the link extension feature to search for the drawers of these two key members.

    Select these two key members, and click **Link Extension**. Set **Link Type** to **Invoicing** to query the drawers of these two key members.

    Results: Analysts found a group of drawers by analyzing one of the key members.

    Inferences made by analysts: This company provides both manufacturing and trading services. It provides trading services for domestic manufacturers from various industries, including the electromechanics industry, electronics industry, moulding industry, and the food industry. This is a suspicious issue of tax cheating.

13. Analysts checked the behavior details of these drawers and the key member.

    Select this key member and the group of drawers. Click **Behavior Chronology**, and then click the **Details** tab. Set **Link** to **Invoicing** to view the invoicing details.

    Results: The proportion of sales and export volumes are both around 10%, which is relatively even.

    Inferences made by analysts: The proportion of orders and exports are too even, which shows obvious human manipulation.

14. Analysts used **Common Neighbors** to check whether these drawers have shared objects, for example, WIFI.

    Select the drawer nodes, and choose **Analyze** > **Common Neighbors**. Set **Link Type** to **WIFI** to check whether the drawers have a shared WIFI.

    Results: These drawers often connect to WIFI using the same MAC address to log on to the trading platform.

    Inferences made by analysts: This is an obvious abnormal phenomenon. It is very likely that these companies are operated by the same group of people.

15. Analysts used **Common Neighbors** to check whether the drawers and the customers have shared phone numbers.

    Select all drawers and customer nodes, and choose **Analyze** > **Common Neighbors**. Set **Link Type** to **Drawer TEL** and **Customer TEL** to check whether the drawer and the customer have a shared contact number.

    Results: A telephone number was shared by different drawers and customers.

    Inferences made by analysts: Based on the abnormal information found in the relationship network, it is very likely that these groups are cheating on tax rebates.

On-the-spot investigation

According to the results returned from the on-the-spot investigation, all these manufacturers and traders are highly suspicious. The department concerned continued to investigate these groups, and uncovered a tax cheat group related with 13 enterprises. This group registered different roles on the trading platform, made fake invoices and transactions internally, and the amount of tax refunds reported to the trade platform reached 100 million RMB.

8.16. FAQ

1. Q: How can I log on to the system for the first time?

   A: Contact the administrator to obtain the account and the initial password to log on to Graph Analytics. To keep your account secure, modify the password as prompted.

2. Q: Why does the system prompt an error message indicating incorrect user name and incorrect password when I was logging on to Graph Analytics?

   A: If you are unable to log on to the system, check whether the account name and the password you entered are correct, and whether the password is entered in half-width characters, in the correct case, or has any space. If the error persists, contact the administrator.

3. Q: What is the default rule for the simple link extension (double-clicking a link to start a link extension)?

   A: To help you analyze the information quickly, the administrator configures common links in Administration Console and synchronizes them to the double-click operation. These common links must be published by the administrator. If you need to add or remove a double-click link extension, contact the administrator.

4. Q: Why does the system prompt "a maximum of five nodes can be selected" when I select all nodes in the graph area to perform the link lookup analysis?

   A: By default, Graph Analytics supports a maximum of five nodes in the link extension analysis. If you need to analyze more than five nodes, contact the administrator, and the administrator will assess the system scale and make adjustments.

5. Q: Why can't I delete a node by pressing Delete on the keyboard?

   A: Currently, you cannot delete a node by pressing Delete on the keyboard. You can right-click the selected node and click **Delete** in the shortcut menu.

6. Q: How many steps can be undone or rolled back?

   A: A maximum of 20 steps can be rolled back or undone.

7. Q: Why does the Path Analysis button turn gray after I select a node?

   A: Path Analysis is available only after two nodes are selected.

8. Q: After I select a node, why do the Group Analysis button and the Common Neighbors button turn gray?

   A: Group Analysis and Common Neighbors are available only after two or more nodes are selected.

9. Q: Can I configure a new link analysis method discovered by myself while using Graph Analytics?

   A: Currently, links are configured by the administrator. If you need to add a new link analysis method, contact the administrator.

10. Q: Why is there no data available after I click Behavior Analysis, Chronology Analysis, and Details?

    Q: You need to select a node in the graph area before performing these analyses or viewing behavior details.

11. Q: Why are statistics displayed on the right side when no node in the graph area is selected?

    A: If no node is selected, statistics on all nodes are collected by default.

12. Q: How to find the nodes that fall out of the scope of the canvas?

    A: You can enable the thumbnail to locate the nodes, and move the nodes that fall out of the canvas to the visible area.

13. Q: Why are all layout buttons become gray and unavailable?

    A: The layout buttons become available only after you select a node in the graph area.

14. Q: After I select all the nodes in the graph area, why does the system give no response after I click Hierarchical Layout?

    A: In the Hierarchical Layout mode, you need to select one node as the starting point to view the hierarchical layout.

15. Q: Besides the drag button in the toolbar, can I use any shortcut operations to move the canvas?

    A: You can press **Space** on the keyboard to move the canvas.

9.Dataphin

9.1. What is Dataphin?

Dataphin is an intelligent engine for building big data platforms. It is designed to meet the requirements of big data development, management, and application across multiple industries. Dataphin combines technologies and methodologies. It provides all-in-one intelligent data development and management services, including data ingestion, data standardization, data modeling, data asset management, and data services.

Dataphin applies to different computing and storage environments. This enables you to use a single console to process data from various data sources. Dataphin allows you to import data, standardize data production, develop data by data modeling, and create a tag system by extracting tags from entities. You can also generate and manage data assets by using your business data and knowledge. Dataphin also provides multiple types of data services such as table query and intelligent voice search.

9.2. Usage notes

This topic describes the notes on using Dataphin.

To use Dataphin, you must have the necessary knowledge and expertise. This document is intended for:

- Application developers
- Analysts
- Data developers
- Technical architects
- System administrators

To ensure stable running of Dataphin, observe the following limits and recommendations.

Operation	Limit and recommendation
Set the computing engine type	Select a computing engine type and configure the cluster where your computing engine resides. For example, specify the endpoint of the cluster. MaxCompute and Hadoop are the available computing engine types. The system uses the selected computing engine type to support data construction in the specified cluster. Select the computing engine type and configure the computing engine settings based on your computing engine cluster.
Create a data source	• We recommend that you configure an AccessKey pair with administrative privileges for data source management. • One physical database can be used only as one data source.

Operation	Limit and recommendation
Create a project	• When the data source type is MaxCompute, the project name must be the same as your MaxCompute project name. • The project name cannot start with `LD_` or `ld_`. If project names start with LD_ or ld_, project names conflict with business unit names. In this case, errors may occur during table query because the system cannot differentiate between a logical table and a physical table. In Dataphin, when you query a logical table, you must prefix the table name with the corresponding business unit name. When you query a physical table, you must prefix the table name with the corresponding project name.
Configure the computing engine	• If a physical database is configured as a computing engine for a project, we recommend that you do not add, delete, or modify data in the database in the consoles of other cloud services. • We recommend that you do not configure computing engines of the same type that reside in different clusters for a project.
Process data	Computing engines can read data only from their own clusters.
Perform data standardization and data modeling	• We recommend that you exercise caution when you name data standardization objects and logical tables. Set names in lowercase for data standardization objects and make sure that the names are valid and easy to read. The names cannot be changed when they have downstream dependencies. • Use abbreviations whenever possible to avoid data production errors. Errors may occur if the length of a field name exceeds the limit that is imposed by the database.
Run an ad hoc query	When you query a logical table, prefix the table name with the corresponding business unit name. When you query a physical table, prefix the table name with the corresponding project name.
Extract the data value	We recommend that you set IDs based on user information to ensure an accurate match between an ID and a user.

9.3. Quick start

9.3.1. Instructions for system administrators

To prepare Dataphin for use, system administrators and deployment engineers must make sure that the environment is ready and required user roles are created.

Procedure

1. Make sure that the following environment and resources are ready:
   - Apsara Infrastructure Management Framework is deployed. The Apsara Stack console is accessible.
   - MaxCompute, Object Storage Service (OSS), DAuth, Server Load Balancer (SLB), Elastic

Compute Service (ECS), three physical machines, and PostgreSQL database resources are available.

> **Note** The PostgreSQL database in your environment must be accessible. This PostgreSQL database can be deployed on an ApsaraDB RDS for PostgreSQL instance or an ECS instance:
>
> - ApsaraDB RDS for PostgreSQL: A 6U model is required for Apsara Stack V3.7.0 and earlier versions. For versions later than V3.7.0, a 7U model is required. We recommend that you do not use miniRDS because it will soon become obsolete.
> - PostgreSQL on ECS: Dataphin provides this configuration in Apsara Stack V3.7.0 and later versions. You can obtain this baseline configuration if you purchase one of these versions. If you purchase a version earlier than V3.7.0, we recommend that you contact the Dataphin team to obtain specific ECS resources and deploy PostgreSQL in a non-standard way.

2. Obtain the following information about the computing engine cluster:
   - MaxCompute endpoint: This information is required when you set the computing engine type.
   - AccessKey ID and AccessKey secret that are used to create the Dataphin_Meta project in MaxCompute: This MaxCompute project is used to compute and store MaxCompute metadata.

   > **Note**
   >
   > After Dataphin and MaxCompute are deployed, a MaxCompute project is created to obtain MaxCompute metadata in the Apsara Stack environment. System administrators and deployment engineers must verify that the Dataphin_Meta project exists and obtain the AccessKey ID and AccessKey secret that are used to create this MaxCompute project. If the Dataphin_Meta project does not exist, contact deployment engineers to create one and grant Dataphin the permission to obtain MaxCompute metadata.

3. Create user roles. After Dataphin is deployed, the account system provides the following three types of user roles in the Apsara Stack environment:
   - O&M super administrator: an independent metadata management tenant of the system. The O&M super administrator can obtain and parse the metadata of your cluster. Each system has only one O&M super administrator. Contact deployment engineers to obtain the username and password of the O&M super administrator.

     > **Note** System administrators must keep the username and password of the O&M super administrator confidential, and exercise caution after logon as the O&M super administrator.

   - Super administrator: a tenant who performs construction operations. For example, the super administrator can manage users and also design and build the architecture of a specific business data system. In the Apsara Stack environment, the role of a department account or Apsara Stack tenant account is the super administrator.
   - Common user: a member of a department. A common user performs development operations, including the detailed design of a specific business data system. In the Apsara

Stack environment, a RAM user can be added under a department account. A RAM user can be added as a tenant member of the super administrator.

> **Note**
>
> Each user can belong to only one tenant. A user from one department cannot be added as a tenant member of another department. We recommend that each Dataphin system be used only for one department, namely, one tenant.

9.3.2. Log on to the Dataphin console

This topic describes how to log on to the Dataphin console in the Apsara Stack Cloud Management (ASCM) console.

Prerequisites

- Before logging on to the ASCM console, make sure that you have obtained the IP address or domain name of the ASCM console from the deployment personnel. The URL used to access the ASCM console is in the following format: https://[IP address or domain name of the ASCM console].
- We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to access the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password for logging on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username as prompted. Due to security concerns, your password must meet the minimum complexity requirements: The password must be 8 to 20 characters in length and must contain at least two of the following character types: uppercase letters, lowercase letters, digits, and special characters such as exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products** > **Big Data** > **Dataphin**.
5. Select the target organization and region. Then, click **Dataphin**.

9.3.3. Prepare for using Dataphin

This topic describes the preparations that you must complete before you use Dataphin.

Prepare items except data sources

1. Specify the endpoint of the computing engine cluster. For more information, see Set the computing engine type.
2. Create a project and configure the sandbox whitelist. For more information, see Create

projects.

3. Create a data domain named test_dataphin. For more information, see Create a data domain.
4. Create a table named dataphin_test as the destination table of data synchronization. For more information, see Create a destination table for data synchronization. You can execute the following SQL statement to create the destination table:

```
CREATE TABLE IF NOT EXISTS `datax_test` (order_id bigint comment 'Order ID',`area` string comme
nt 'Region',province string comment 'Province',city    string comment 'City',product_type string com
ment 'Type',order_name string comment 'Customer name',report_date datetime comment 'Date',or
der_amt double comment 'Sales amount')PARTITIONED BY (  `ds` STRING);
```

5. Add members to the project. For more information, see Add members. In this tutorial, add two members and set their roles to **Developer** and **Project Administrator**, respectively.

Prepare data sources

In this tutorial, statistics are collected on the total daily sales of office supplies and technical products in each province.

1. Create data sources. In this tutorial, create an ApsaraDB RDS for MySQL instance named dataphin. For more information, see Quick start in *ApsaraDB for RDS User Guide*.
2. Download the company_sales_recrod_copy table to be used in this tutorial.
3. Upload the company_sales_record_copy table to the ApsaraDB RDS for MySQL instance that you create.

9.3.4. Ingest data

This topic uses ApsaraDB RDS for MySQL as an example to describe how to connect a data source to Dataphin.

Prerequisites

A data source and a project are prepared. For more information, see Prepare for using Dataphin.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Planning** in the top navigation bar.
3. On the Planning page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source to be connected to Dataphin. Select **MYSQL**.
Name	The name of the data source. Set the value to dataphin.
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The Java Database Connectivity (JDBC) URL that you can use to connect to the data source. Enter a URL in the format of `jdbc:mysql://RDS ID.mysql.rds.aliyuncs.com:3306/dataphin`. Replace RDS ID with the ID of the ApsaraDB RDS for MySQL instance that you want to connect to Dataphin.
Username	The username that you can use to connect to the data source. Set the value to dataphin.
Password	The password that you set for the dataphin user when you create the ApsaraDB RDS for MySQL instance.

6. Click **Test Connection**.
7. After the data source passes the connectivity test, click **OK**.

9.3.5. Integrate data

This topic describes how to synchronize data from a data source to your project, including the procedure for creating and configuring a sync task.

Prerequisites

A data source is connected to Dataphin. For more information, see Ingest data.

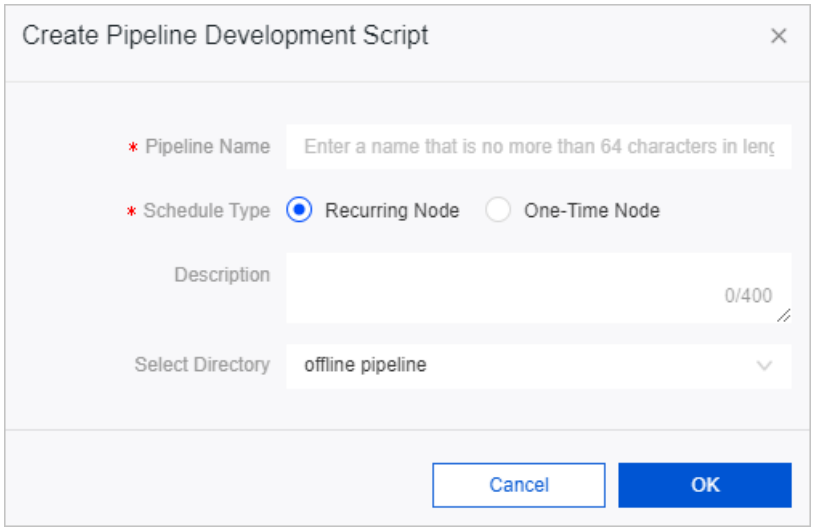
Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. Move the pointer over **Develop** in the top navigation bar and select **Integrated**.
5. Create an offline migration pipeline.

i. On the **Integrated** page, open the **Create Pipeline Development Script** dialog box by using one of the following methods:

- In the Script section, move the pointer over the icon next to Script and select **Offline Single Pipeline**.
- In the left-side navigation pane, move the pointer over the icon next to the project name and select **Offline Single Pipeline**.

ii. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Select **One-Time Node**.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides. Default value: **offline pipeline**.

iii. Click **OK**.

6. Configure the offline migration pipeline.

i. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.

ii. Click the icon before **Input**. Drag the **MySQL** icon to the pipeline canvas on the left.

The MYSQL_1 component appears on the canvas.

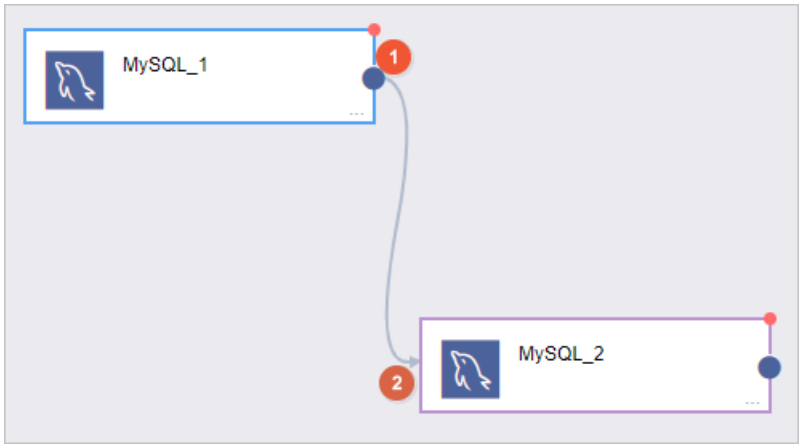
iii. Right-click MYSQL_1 and select **Configure Attributes**. In the MySQL Input Configuration dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ■ The data source is of the MySQL type. ■ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for the permission. For more information, see Data source permissions. You can also click the icon next to **Data Source** to go to the Planning page and connect a data source to Dataphin.
Source Table	The mode for synchronizing data. Select **Single Table**.
Table	The name of the source table. In this example, select company_sales_record_copy.
Split Key	Optional. The shard key of the source table.
Input Filtering	Optional. The filter condition for input fields.
Output Fields	The output fields to be generated based on the input configuration.

iv. Click **OK**.

v. Click the icon before **Output**. Drag the **Maxcompute** icon to the pipeline canvas on the left. The Maxcompute_1 component appears on the canvas.

vi. Drag a directed line from **MySQL_1** to **Maxcompute_1** to establish a relationship between the two components, as shown in the following figure.



vii. Right-click Maxcompute_1 and select **Configure Attributes**. In the Maxcompute Output Configuration dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Data Source	The data source of the component. Select a data source that is bound to your project.
Table	The name of the destination table. In this example, select dataphin_test.
Loading Policy	The policy for writing data to the destination table. Select **Append Data**.
Parse Solution	Optional. The operations before and after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated.
Quick Mapping	The GUI element that allows you to specify mappings between the input and output fields. To specify name-based mappings between the input and output fields, perform the following steps: a. Move the pointer over the icon next to **Quick Mapping**. b. Select **Name-based Mapping** from the drop-down list. c. In the **Notification** message, click **OK**.

viii. Click **OK**.

7. In the upper-left corner of the pipeline configuration tab, click **Preview** to check whether the pipeline can be run.
   - If some variables such as bizdate are configured in the components, set relevant parameters as required and click **OK** to preview the running result of the pipeline.
   - If no variable is configured in the components, you can directly preview the running result of the pipeline.
8. In the upper-left corner of the pipeline configuration tab, click **Run** to synchronize data from the data source of the MYSQL_1 component to the data source that is bound to your project.
   - If some variables such as bizdate are configured in the components, set relevant parameters as required and click **OK** to run the pipeline.
   - If no variable is configured in the components, you can directly run the pipeline.
9. Save, submit, and then publish the offline migration pipeline.
   i. Click the icon in the upper-right corner to save the pipeline.

ii. Click the icon in the upper-right corner to submit the pipeline.When you submit the pipeline, Dataphin checks whether you have the following permissions:
   - Read permission on the data source in each input component
   - Write permission on the data source in each output component

iii. (Optional)Publish the pipeline.
   - If the current project is in Dev mode, publish the pipeline to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the pipeline after you submit it.

9.3.6. Define data standardization objects

Based on the methodology of dimensional modeling, you can build a bus matrix, divide data domains, and then define business processes, dimensions, atomic metrics, business filters, statistical periods, and derived metrics. This topic describes how to define data standardization objects.

Sample data

In this tutorial, statistics are collected on the total daily sales of office supplies and technical products in each province. The following table describes the fields that are selected from the company_sales_record_copy table for analysis.

Field	Description
order_id	The order ID.
area	The region.
province	The province.
city	The city.
product_type	The type of the product.
customer_name	The name of the customer.
report_date	The date of the order.
order_amt	The sales amount of the order.

Terms

- Dimension

  A dimension, which serves as the basis of measurement, is composed of a type of business attributes. A dimension can also be called an entity object. When you divide data domains and build a bus matrix, you must define dimensions based on the analysis of business processes. In this example of analyzing the total daily sales of office supplies and technical products in each province in a year, province is used as the dimension for data modeling.

- Business process

A business process refers to a business event of an enterprise, such as order placement, payment, and refund. In most cases, a business process is an indivisible event. This tutorial uses order placement as the business process whose ID is order_pay.

- Atomic metric

  An atomic metric is the measure for a business event. It is an indivisible metric and has a specific business meaning. You must define the `measure and business event` for an atomic metric. For example, the payment amount is an atomic metric, where payment is the event and amount is the measure. In this tutorial, the atomic metric sum(order_amt) is created based on the sum of the measure order_amt.

- Business filter

  A business filter defines the scope of a business. A business filter is unique within a business unit and belongs to only one source logical table. The computing logic of a business filter is defined based on the fields of the source logical table model. This ensures that all metrics are created in a uniform and standardized manner. In this tutorial, `product_type='technical products'` and `product_type='office supplies'` are used as business filters.

- Derived metric

  A derived metric is composed of the `atomic metric, business filter, statistical period, and dimension (statistic granularity)`. A derived metric is an atomic metric that is constrained by the statistical business scope. This tutorial takes the total sales amount of office supplies and technical products of the last day in each province as the derived metric, province as the dimension, order placement as the business process, the total sales amount sum(order_amt) as the atomic metric, technical products and office supplies as business filters, and the last day as the statistical period.

9.3.7. Create data modeling objects

This topic describes how to create data modeling objects.

Prerequisites

- A data source and a project are prepared. For more information, see Prepare for using Dataphin.
- The data source is connected to Dataphin. For more information, see Ingest data.
- Data synchronization is completed. For more information, see Integrate data.
- Data standardization objects are defined. For more information, see Define data standardization objects.

Create a dimension

1. Log on to the Dataphin console.
2. Go to the **Dimensions** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. (Optional)On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

iii. In this example, select the project that you create.

iv. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

v. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon.

3. In the **Dimensions** section, click the icon in the upper-right corner.

4. On the **Create Dimension** tab, set the parameters as required.

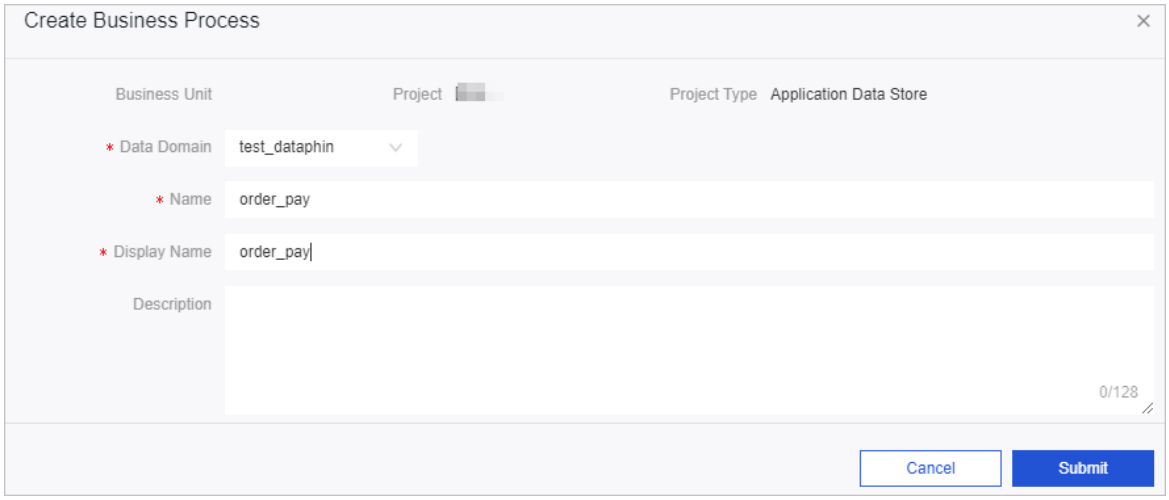
Section	Parameter	Description
Basic Dimension Information	Data Domain	The data domain of the dimension to be created. Select test_dataphin.
	Dimension Name	The name of the dimension. Enter province.
	Dimension Display Name	The display name of the dimension. Enter province.
	Dimension Description	The description of the dimension.
Dimension Logic	Primary Key Name	The name of the primary key. Enter province.
	Primary Key Display Name	The display name of the primary key. Enter province.
	Primary Key Type	The data type of the primary key. Select STRING.

Section	Parameter	Description
	Primary Key Computing Logic	The computing logic of the primary key. Dataphin allows you to define the primary key computing logic of a dimension by writing SQL statements. i. Click **Example** next to **Primary Key Computing Logic** to view an example of SQL statements. Enter the following SQL statement in the SQL editor: `select` `province` `from` `dataphin_test` ii. Click **Code Check** to verify whether the SQL statement that you entered is valid.

5. Save and submit the dimension.
   i. Click the icon in the upper-right corner to save the dimension.
   ii. Click the icon in the upper-right corner to submit the dimension.
6. View the corresponding logical dimension table.
   i. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon.After you create a dimension, Dataphin automatically generates a logical dimension table for the dimension. The name of the generated logical dimension table is the same as the dimension name that you defined.
   ii. In the Logical Dimension Tables section, use the dimension name that you defined to search for the corresponding logical dimension table. Alternatively, search for the logical dimension table dim_province in the data domain test_dataphin.

Create a business process and a logical fact table

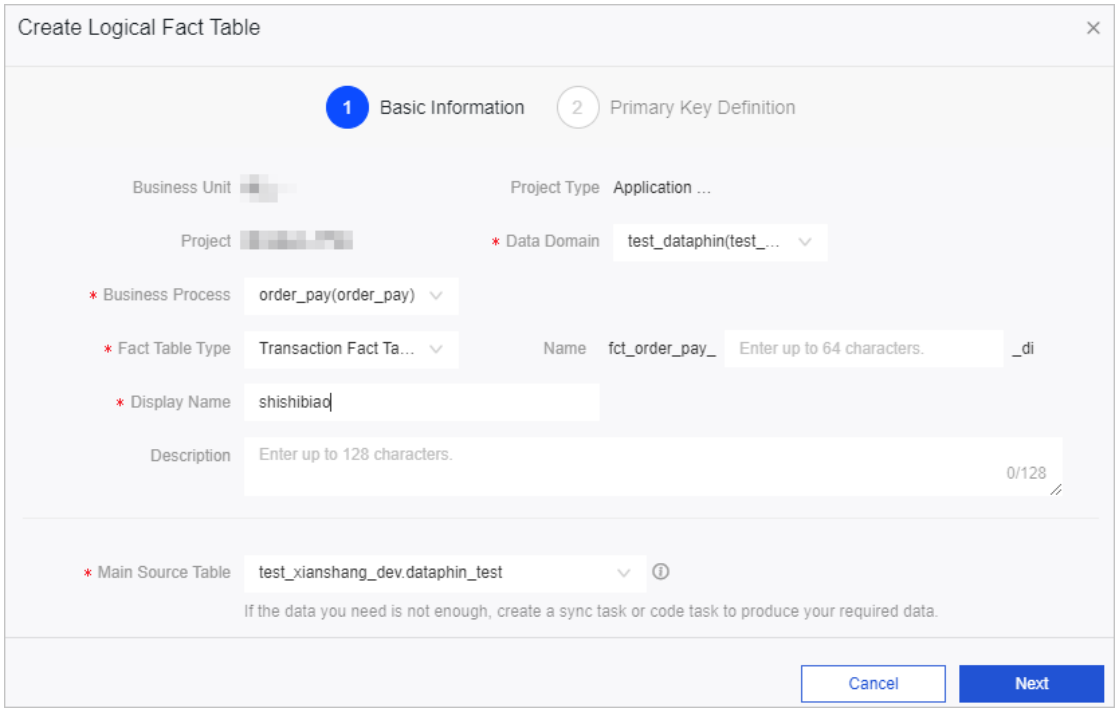
1. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon.
2. In the **Business Processes** section, click the icon in the upper-right corner.
3. In the **Create Business Process** dialog box, set the parameters as required.



Parameter	Description
Data Domain	The data domain of the business process to be created. Select test_dataphin.
Name	The name of the business process. Enter order_pay.
Display Name	The display name of the business process. Enter order_pay.

4. Click **Submit**.
5. In the dialog box that appears, enter your comments.
6. Click **OK**.
7. Create a logical fact table.
   i. Click **order_pay** in the Business Processes section. On the View Attributes tab, you can view the details about the business process.
   ii. Move the pointer over the icon and select **Create Logical Table**.

iii. In the **Create Logical Fact Table** dialog box, set the parameters as required.



Parameter	Description
Business Process	The business process that corresponds to the logical fact table to be created. Select order_pay (order_pay).
Fact Table Type	The type of the logical fact table. Select **Transaction Fact Table**.
Display Name	The display name of the logical fact table. Enter shishibiao.
Main Source Table	The main source table of the logical fact table. Select Project name.dataphin_test. In this example, select test_xianshang_dev.dataphin_test.

iv. Click **Next**.

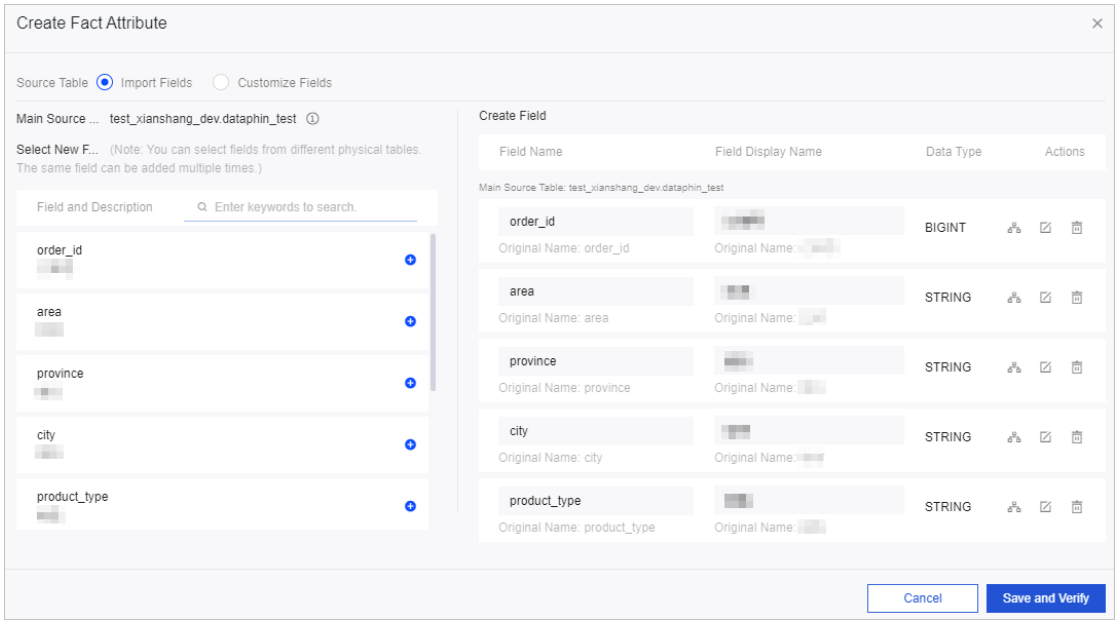
v. In the **Primary Key Definition** step, set the **Set Primary Key** parameter to **No**.

vi. Click **Submit**.

8. Configure the logical fact table.

i. On the configuration tab of the **fac_order_pay_df** logical fact table that appears, click **Add Measure** in the Central Table section.

ii. In the **Create Measure** dialog box, set the parameters as required.

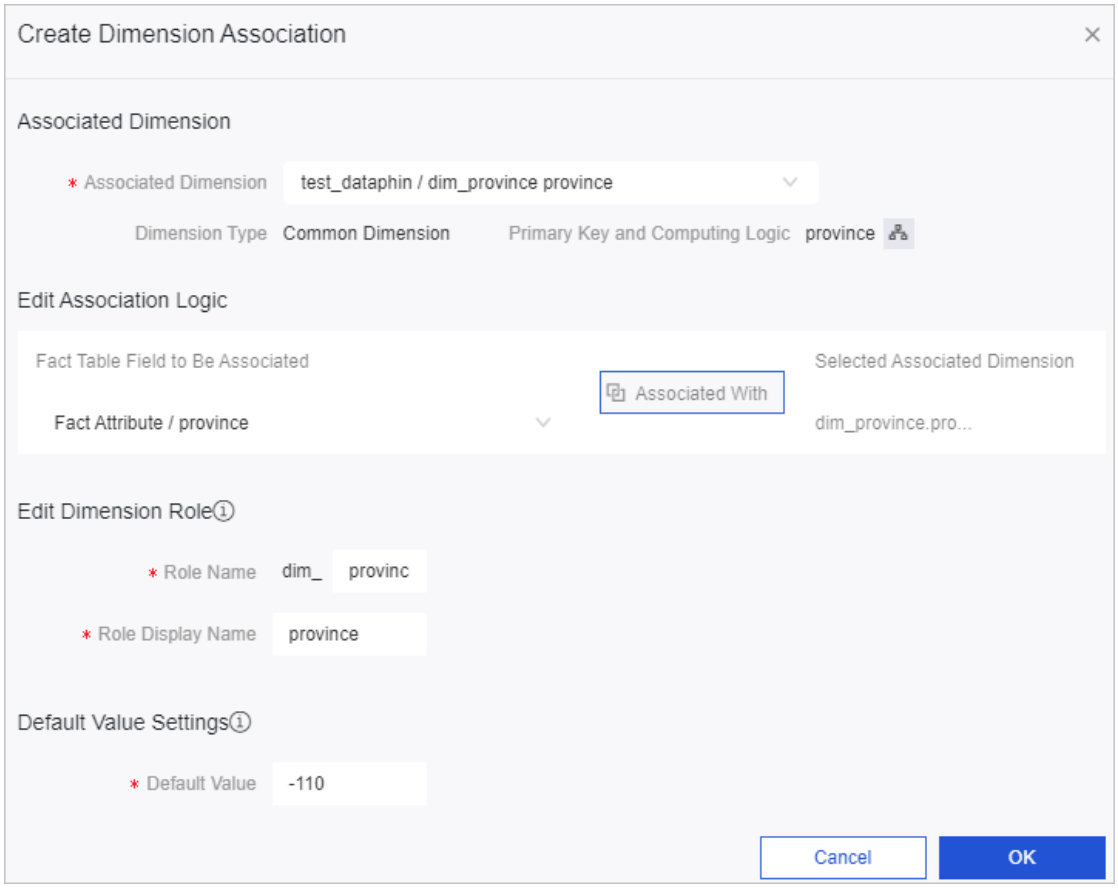


Parameter	Description
Source Table	Specifies whether to import fields from the source table or use SQL statements to reference fields. Select **Import Fields**.
Create Field	The field to be added. Add a field by performing the following steps: a. Click the icon next to a field name in the Select New Fields section. b. Enter a name in the **Field Display Name** field.

iii. Click **Save and Verify**.

iv. On the configuration tab of the fac_order_pay_df logical fact table, click **Create Dimension Association** in the Central Table section.

v. In the **Create Dimension Association** dialog box, set the parameters as required.



- **Associated Dimension**: Select the logical dimension table dim_province in the data domain test_dataphin.
- **Edit Association Logic**: Select province under Measure.
- Use the default values for other parameters.

vi. Click **OK**.

9. Save and submit the logical fact table.

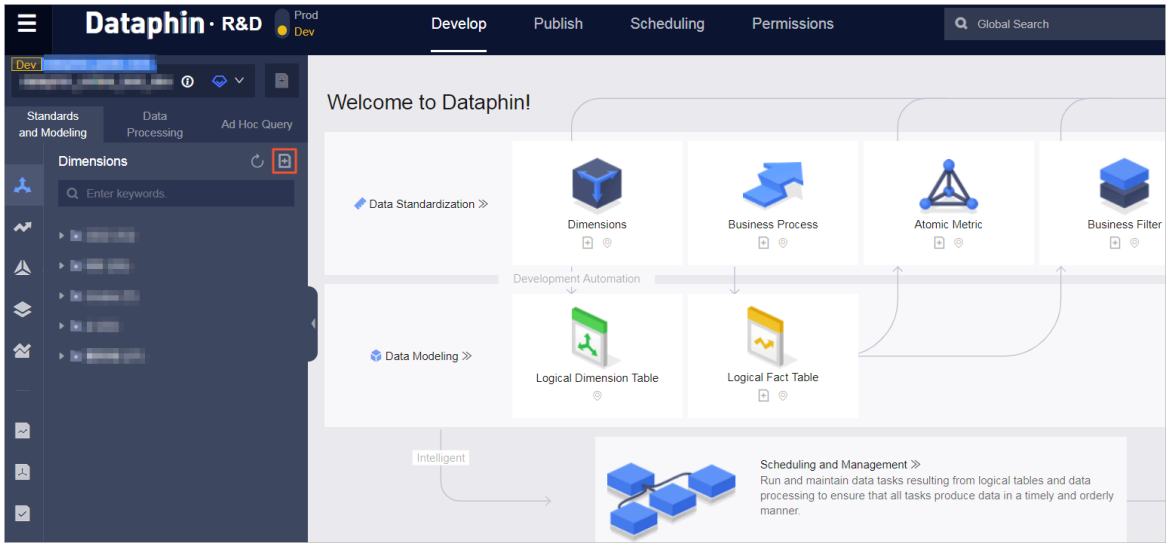
   i. Click the icon in the upper-right corner to save the logical fact table.

   ii. Click the icon in the upper-right corner to submit the logical fact table.

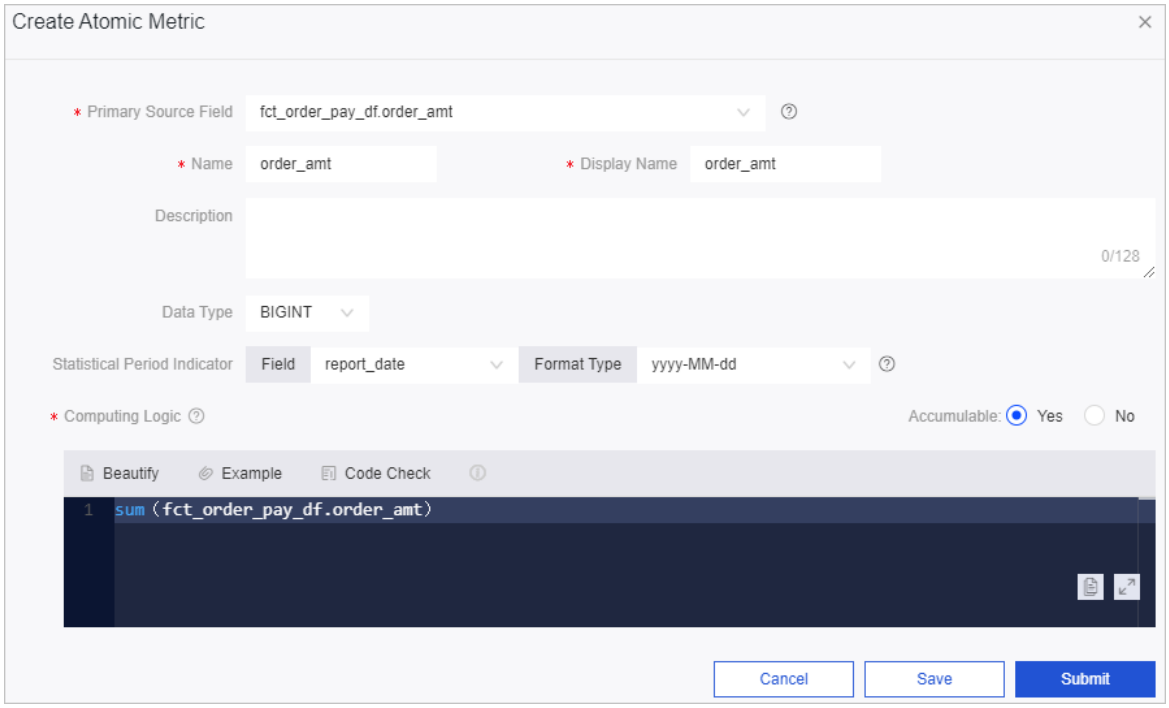
Create an atomic metric

1. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon.

2. In the **Atomic Metrics** section, click the icon in the upper-right corner and select **Create Atomic Metric**.



3. On the **Create Atomic Metric** tab, select fct_order_pay_df from the **Source Table** drop-down list. In the Atomic Metrics section, click **Create Atomic Metric**.
4. In the **Create Atomic Metric** dialog box, set the parameters as required.



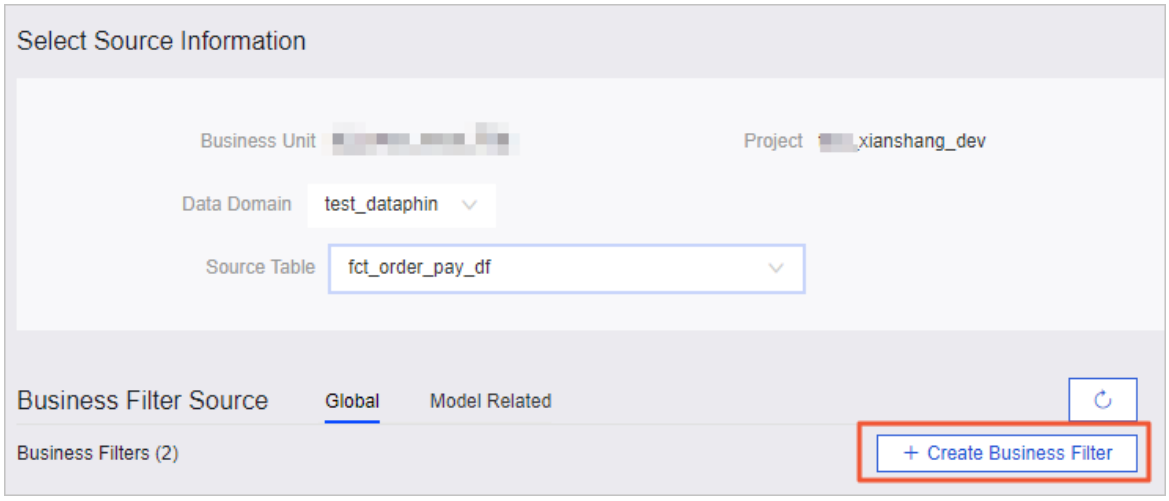
Parameter	Description
Primary Source Field	The source field of the atomic metric to be created. Select fct_order_pay_df.order_amt.
Name	The name of the atomic metric. Enter order_amt.
Display Name	The display name of the atomic metric. Enter order_amt.
Data Type	The data type of the atomic metric. Select BIGINT.

Parameter	Description
Field	The field of the statistical period. Select report_date.
Computing Logic	The computing logic of the atomic metric. Enter `sum(fct_order_pay_df.order_amt)`.

5. Click **Code Check** to verify whether the SQL statement that you entered is valid.
6. After the SQL statement passes the code check, click **Submit**.
7. In the dialog box that appears, enter your comments.
8. Click **OK**.

Create a business filter

1. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon.
2. In the **Business Filters** section, click the icon in the upper-right corner.
3. On the **Create Business Filter** tab, select `fct_order_pay_df` from the **Source Table** drop-down list. In the Business Filter Source section, click **Create Business Filter**.



4. In the **Create Business Filter** dialog box, set the parameters as required.

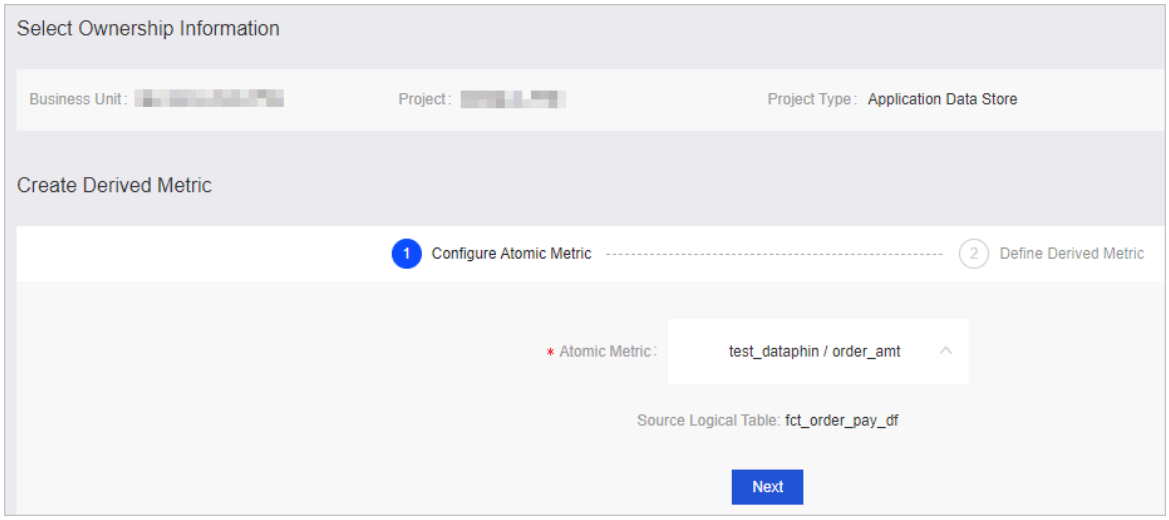
Parameter	Description
Primary Source Field	The source field of the business filter to be created. Select fct_order_pay_df.product_type.
Name	The name of the business filter. Enter product_type_bangongyongpin.
Display Name	The display name of the business filter. Enter product_type_office supplies.

Parameter	Description
Computing Logic	The computing logic of the business filter. Enter `fct_order_pay_df.product_type='office supplies'` .

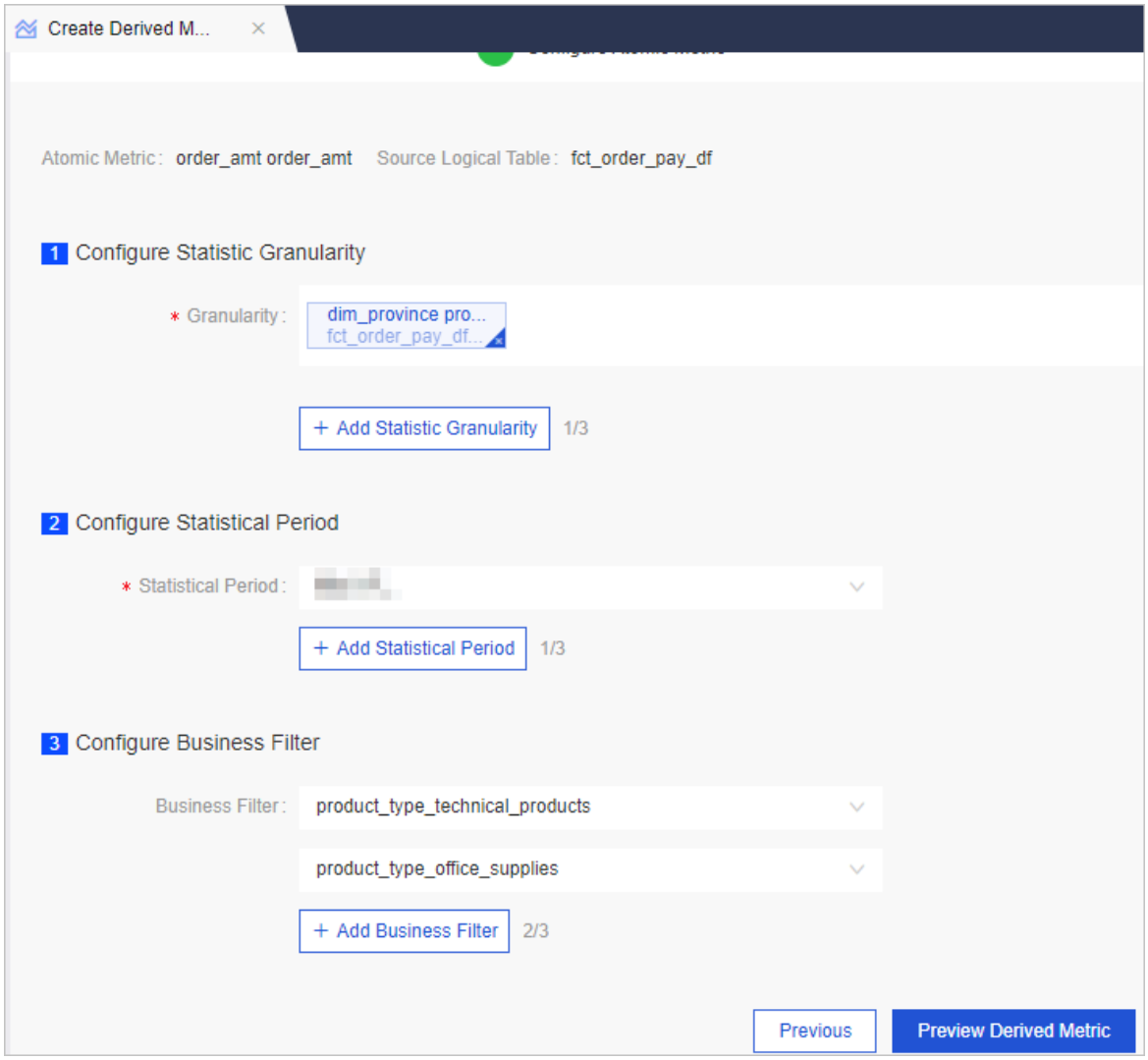
5. Click **Code Check** to verify whether the SQL statement that you entered is valid.
6. After the SQL statement passes the code check, click **Submit**.
7. In the dialog box that appears, enter your comments.
8. Click **OK**.

Create a derived metric

1. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon.
2. In the **Derived Metrics** section, click the icon in the upper-right corner.
3. On the **Create Derived Metric** tab, select order_amt under test_dataphin from the **Atomic Metric** drop-down list.



4. Click **Next**.
5. In the **Define Derived Metric** step, set the parameters as required.



Parameter	Description
Granularity	The statistic granularity of the derived metric to be created. Select fct_order_pay_df.dim_province under dim_province province.
Statistical Period	The statistical period of the derived metric. Select the last day.
Business Filter	The business filter of the derived metric. Select product_type_office supplies and product_type_technical products.

6. Click **Preview Derived Metric**.
7. In the **Change Derived Metric** section, confirm the parameter settings and click **Submit**.
8. Submit the derived metric.
   i. Click **Submit** to submit the derived metric.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.

9.3.8. Generate retroactive data

To verify that code and tasks are valid, you can initiate a retroactive data generation task and verify the data. This topic describes how to generate retroactive data for logical table conversion tasks and sync tasks.

Prerequisites

Data modeling objects are created. For more information, see Create data modeling objects.

Context

After you create and submit a dimension, a logical fact table, and a logical aggregate table, Dataphin automatically generates the corresponding logical table conversion tasks, for example, dim_province_core_od001_v1, fct_order_pay_df_od000_v1, and dws_province_od000_v1. You can use the same method to generate retroactive data for these logical table conversion tasks. This topic describes how to generate retroactive data for the logical table conversion task that is generated after you create a dimension, namely, dim_province_core_od001_v1.

Generate retroactive data for a logical table conversion task

1. Log on to the Dataphin console.
2. Go to the **Logical Table Conversion Tasks** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select the project that you create.
   iii. In the top navigation bar, click **Scheduling**.
   iv. In the left-side navigation pane, click the **Global Management** tab.
   v. On the **Global Management** tab, move the pointer over the left-side navigation submenu and click the icon.
   vi. In the **Recurring Tasks** section, click the **Logical Table Conversion Tasks** tab.
3. Click the logical table conversion task dim_province_core_od001_v1.
4. In the directed acyclic graph (DAG), right-click the current node and select **Generate Retroactive Data**.
5. In the **Generate Retroactive Data** dialog box, set the **Data Timestamp** parameter to 2019-10-27~2019-10-27. Set the **Select Downstream Nodes** parameter to **No**.

   > **Note** The system automatically generates a value for the **Instance Name** parameter when you open the **Generate Retroactive Data** dialog box. You can change the value.

6. Click **OK**.
7. View the running result of the retroactive data generation instance.
   i. On the Global Management tab, move the pointer over the left-side navigation submenu and click the icon.

ii. In the Retroactive Data Generation Instances section, click the retroactive data generation instance. In the DAG, right-click the current node and select **View Operations Log**.

iii. In the **Operations Log** pane, view the operational logs of the instance.

Generate retroactive data for a sync task

1. On the **Global Management** tab, move the pointer over the left-side navigation submenu and click the icon.
2. In the **One-time Tasks** section, click the one-time task **dataphin**.
3. On the page that appears, click **Run** in the upper-right corner.
4. In the **Run** dialog box, set the **Data Timestamp** parameter to `2019-10-27`.
5. Click **OK**.
6. View the running result of the one-time instance.
   i. On the Global Management tab, move the pointer over the left-side navigation submenu and click the icon.
   ii. In the **One-time Task Instances** section, click the instance that is generated for the one-time task **dataphin**.
   iii. On the page that appears, click **View Operational Log** in the top navigation bar. On the Operational Log page, view the operational logs of the instance.

9.3.9. Verify data

This topic describes how to use an ad hoc query task to verify that your expected data is generated.

Prerequisites

Retroactive data is generated for the required logical table conversion tasks and sync tasks. For more information about how to generate retroactive data, see Generate retroactive data.

Context

The ad hoc query feature allows you to perform theme-based data query. MaxCompute SQL statements can be executed when MaxCompute is selected as the computing engine type of Dataphin. Dataphin automatically identifies and switches the SQL type based on the computing engine type that you set.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Ad Hoc Query** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select the project that you create.
   iii. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.

3. In the **Ad Hoc Query** section, click the icon next to **Ad Hoc Query**.
4. In the **Create Item** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the ad hoc query task to be created. Enter dim_province.
Description	The description of the ad hoc query task.
Select Directory	The folder where the ad hoc query task resides.

5. Click **OK**.
6. On the **Code Editor** tab of the ad hoc query task, enter `select * from dataphin_test where ds='20191027'`.
7. Save and run the ad hoc query task.
   i. Click the icon in the upper-right corner to save the ad hoc query task.
   ii. Click the Run icon in the upper-right corner to run the ad hoc query task.
8. After the SQL statement is executed, view the running result on the **Result** tab.

	order_id	area	province	city	product_type	customer_name	report_date	order_amt	ds
1	13729				office supplies		2013-01-01 00:00:00	872.48	20191027
2	28774				office supplies		2013-01-01 00:00:00	180.36	20191027
3	37537				technical products		2013-01-02 00:00:00	4083.19	20191027
4	37537				technical products		2013-01-02 00:00:00	4902.38	20191027
5	37537				office supplies		2013-01-02 00:00:00	1239.06	20191027
6	44069				technical products		2013-01-02 00:00:00	137.63	20191027

9.3.10. Publish data

This topic describes how to publish data from the development environment to the production environment for scheduling.

Prerequisites

The data to be published is verified. For more information, see Verify data.

Context

You can publish data only when the data is generated in a project in Dev mode.

- If data is generated in a project in Basic mode, relevant tasks can be scheduled in the production environment after you submit the data.
- If data is generated in a project in Dev mode, you must publish the data to the corresponding project in Prod mode for scheduling after you submit the data. The data to be published may be data standardization and data modeling objects or data processing tasks.

Procedure

1. Log on to the Dataphin console.

2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the Develop page, click the icon next to the project name in the upper-left corner and select the project that you create.
4. In the top navigation bar, click **Publish**.
5. Publish data standardization and data modeling objects.
   i. On the **Objects to Publish** page, click the **Standards and Modeling** tab.
   ii. On the **Standards and Modeling** tab, select the objects that you develop in this tutorial.
   iii. Click **Publish** in the lower part of the page.
   iv. In the **Publish** dialog box, set the **Publishing Name** parameter and click **OK**. You can use the default value of Publishing Name.
6. Publish batch processing tasks.
   i. On the **Objects to Publish** page, click the **Batch Processing** tab.
   ii. On the **Batch Processing** tab, select the sync task that you develop in this tutorial.
   iii. Click the icon in the **Actions** column.
   iv. Alternatively, click **Publish** in the lower part of the page.
   v. In the **Publish** dialog box, set the **Publishing Name** parameter and click **OK**. You can use the default value of Publishing Name.
7. View publishing records.
   i. In the left-side navigation pane, click **Publishing History**. The Publishing History page appears.
   ii. Click the **Standards and Modeling** tab and view publishing records of data standardization and data modeling objects.
   iii. Click the **Batch Processing** tab and view publishing records of data processing tasks.

9.3.11. Verify the scheduling result

This topic describes how to verify that the scheduling result meets your expectations.

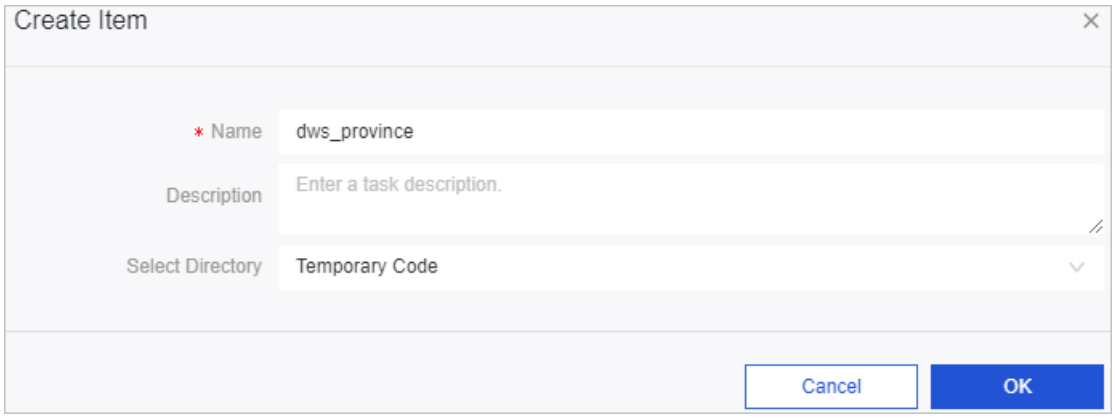
Context

After you publish data to the production environment, relevant tasks will not be run until the next day. You can simulate the task scheduling in the development environment to view the scheduling result.

Procedure

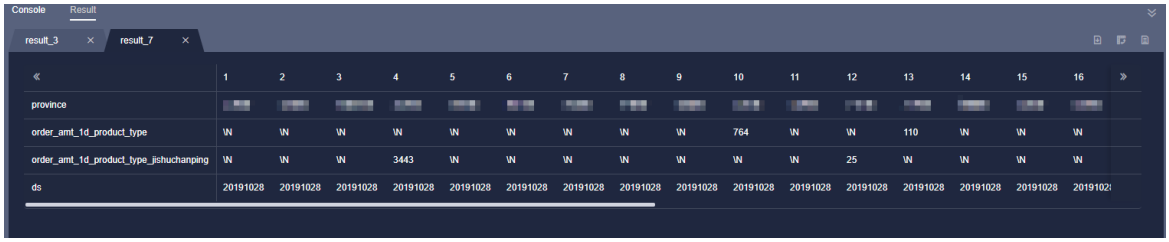
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. Go to the **Global Management** tab.
   i. On the Develop page, click the icon next to the project name in the upper-left corner and select the project that you create.
   ii. In the top navigation bar, click **Scheduling**. The **Global Management** tab appears.

4. Generate retroactive data for one-time tasks and logical table conversion tasks of logical dimension tables, logical fact tables, and logical aggregate tables. For more information, see Generate retroactive data.
5. Go to the **Ad Hoc Query** tab.
   i. In the top navigation bar, move the pointer over **Develop** and select **Develop**. The **Develop** page appears.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.
6. Create an ad hoc query task.
   i. On the **Ad Hoc Query** tab, click the icon.
   ii. In the **Create Item** dialog box, set the parameters as required.



Parameter	Description
Name	The name of the ad hoc query task to be created. Enter dws_province.
Description	The description of the ad hoc query task.
Select Directory	The folder where the ad hoc query task resides.

   iii. Click **OK**.
   iv. On the **Code Editor** tab of the ad hoc query task, enter `select * from dataphin_test where ds='20191027'` .
7. Save and run the ad hoc query task.
   i. Click the icon in the upper-right corner to save the ad hoc query task.
   ii. Click the Run icon in the upper-right corner to run the ad hoc query task.
8. After the SQL statement is executed, view the running result on the **Result** tab.



9.4. Management Center

9.4.1. Initialize metadata

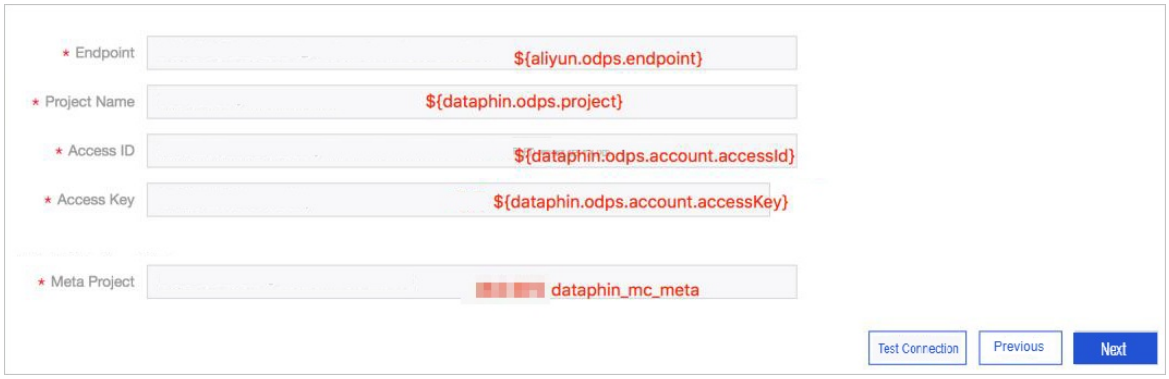
After the compute cluster information is obtained, you can log on to the Dataphin console as the operations and maintenance (O&M) super administrator to configure the metadata warehouse. This topic uses Apsara Stack Dataphin whose computing engine type is MaxCompute as an example.

Procedure

1. Log on to the Dataphin console as the O&M super administrator. For more information about the O&M super administrator account and password, see Instructions for the system administrator.

   > **Note** System administrators must keep the account and password of the O&M super administrator strictly confidential and perform operations with caution after logging on as the O&M super administrator.

2. On the Dataphin homepage, click **Management Center** in the top navigation bar. In the left-side navigation pane, click **Metadata Warehouses Configuration**. On the page that appears, click **Start**.
3. In the Select Computing Engine step, select a computing engine type and click **Next**. In this example, select **MaxCompute v1.1**.
4. In the Parameter Configuration step, set the parameters correctly, as shown in the following figure. Click **Test Connection**. After the test is passed, click **Next**.



5. Perform the subsequent steps as prompted.

> **Note** The subsequent steps, including configuring MaxCompute at the backend, take about 15 minutes. If a message appears to inform you that the subsequent steps are successfully performed, click **Complete**.

9.4.2. Manage members

This topic describes how to add and remove members and configure contact information for members in Dataphin.

Go to the Members Management page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Management Center** in the top navigation bar.
3. On the **Members Management** page, different roles including members and the super administrator have different permissions:
   - If you are a member, you can view the following information of all members: username, display name, and when a member is added to the system. However, you can only view and edit your own mobile number, email address, and webhook URL.
   - If you are the super administrator, you can perform the following operations:
     - View the following information of all members: username, display name, and when a member is added to the system. You can also click the icon in the **Mobile Phone**, **Mailbox**, and **DingTalk Group Chatbot** columns to view the mobile number, email address, and webhook URL of a member.

       After you view the mobile number, email address, and webhook URL of a member, you can click the icon in the **Mobile Phone**, **Mailbox**, and **DingTalk Group Chatbot** columns to hide corresponding information. You can also edit the mobile number, email address, and webhook URL for all members.
     - Click the icon in the headings of the **Mobile Phone**, **Mailbox**, and **DingTalk Group Chatbot** columns to view the mobile numbers, email addresses, and webhook URLs, if configured, of all members.

       After you view the mobile numbers, email addresses, and webhook URLs of all members, you can click the icon in the headings of the **Mobile Phone**, **Mailbox**, and **DingTalk Group Chatbot** columns to hide corresponding information of all members.

Add members

> **Note** Only the super administrator can add members to Dataphin.

1. On the **Members Management** page, click **Add Members** in the upper-right corner.
2. In the **Add Members** dialog box, click the icon next to the **Username** parameter.
3. Select a RAM user and click **OK**.

You can add multiple RAM users as members at a time. You can enter a keyword in the search box to search for RAM users you want to add.

Configure contact information for members

> **Note** The super administrator can configure contact information for one or more members at a time. Members can configure only their own contact information.

To configure contact information for one or more members at a time, perform the following steps:

- i. On the **Members Management** page, select one or more members and click **Edit Members** in the upper-right corner.
  ii. In the **Modify Contact Information** dialog box, set the parameters as required.

Parameter	Description
Mobile Phone	The mobile number of the member. **Note** Only mobile numbers in mainland China can be added.
Mailbox	The email address of the member.
DingTalk Group Chatbot	The webhook URL of the member. Enter `https://oapi.dingtalk.com/robot/send?access_token=713697da1c2a874bc80518e1872e9627d7e79d3015dd5791a1f03589be76ce3f`.

  iii. Click **OK**.
- i. On the **Members Management** page, find the member for whom you want to configure contact information and click the icon in the **Actions** column.
  ii. In the **Modify Contact Information** dialog box, set the parameters as required.

Parameter	Description
Mobile Phone	The mobile number of the member. **Note** Only mobile numbers in mainland China can be added.
Mailbox	The email address of the member.
DingTalk Group Chatbot	The webhook URL of the member. Enter `https://oapi.dingtalk.com/robot/send?access_token=713697da1c2a874bc80518e1872e9627d7e79d3015dd5791a1f03589be76ce3f`. You can move the pointer over the icon to view how to receive alert notifications by using a DingTalk chatbot.

iii. Click **OK**.

Remove members

> **Note** Only the super administrator can remove members from Dataphin.

To remove one or more members at a time, perform the following steps:

- i. To remove multiple members at a time, select the members and click **Delete** in the upper-right corner.
  ii. In the message that appears, click **OK**.
- i. To remove a single member, find the member and click the icon in the **Actions** column.
  ii. In the message that appears, click **OK**.

> **Note** By default, only RAM users who are added as members can be removed from Dataphin. The super administrator cannot be removed.

Synchronize account information from another account system to Dataphin

If the account information is updated in the RAM user list, you can click **Account System Synchronization** on the **Members Management** page to synchronize the updated account information to Dataphin.

9.4.3. Configure the computing engine settings

This topic describes how to configure the computing engine settings of Dataphin.

Context

Dataphin supports the computing engines for both stream processing and batch processing.

- By default, Dataphin sets **Flink** as the computing engine type for stream computing.
- Dataphin supports **MaxCompute** and **Hadoop-2.6** as the computing engine type for batch computing.

Procedure

1. Log on to the Dataphin console as the super administrator.
2. On the Dataphin homepage, click **Management Center** in the top navigation bar.
3. On the page that appears, click **Computing Engine Configuration** in the left-side navigation pane.
4. On the **Computation Configuration** page, select the computing engine type for batch processing.
5. Set the parameters as required.
   - If you select **MaxCompute** as the computing engine type, you must set the **Endpoint** parameter. Click **Verify**. After the setting passes the verification, click **Submit**.
   - If you select **Hadoop-2.6** as the computing engine type, you must set the **NameNode**

parameter. Click **Verify**. After the setting passes the verification, click **Submit**.

9.4.4. Configure the intelligent data engine

9.4.4.1. Data skew optimization

This topic describes how to use the data skew optimization feature of Dataphin to optimize abnormal tasks during scheduling.

Introduction to the data skew optimization feature

The data skew optimization feature in Dataphin is described as follows:

- After the data skew optimization feature is enabled and the first task is submitted, if the duration and processed data volume of the task exceed the specified duration threshold, multiples of the average duration, and multiples of the average data volume, the intelligent data engine starts the preliminary anomaly detection.
- When the task is running, the intelligent data engine detects data for the task to find hotspot data in associated fields. Based on the detected data, the engine refactors the task code to avoid data skew. In addition, the engine kills the previously running task and uses the optimized code to resubmit the task. This optimizes logical table tasks.
- The intelligent data engine records the detected data skew status and uses it to detect data for the next day. If the engine still detects data skew on the next day, it continues to use the optimized code. If no data skew is detected, the engine uses the code without data skew optimization.
- If a rerun task references data tables and fields that are detected by the intelligent data engine but the engine still detects data skew in a field, the engine optimizes the submitted SQL statements based on the detected data. Then, the engine uses the optimized code to rerun the task. If a rerun task references data tables that are not detected by the intelligent data engine, the engine detects data for the task only when the task exceeds the specified thresholds, and then the engine refactors the code based on the detected data. The intelligent data engine needs to process data skew detected during production in tasks and models that it automatically generates.

If you want to use the data skew optimization feature, you must set YARN parameters for the computing engine, as shown in the following figure.

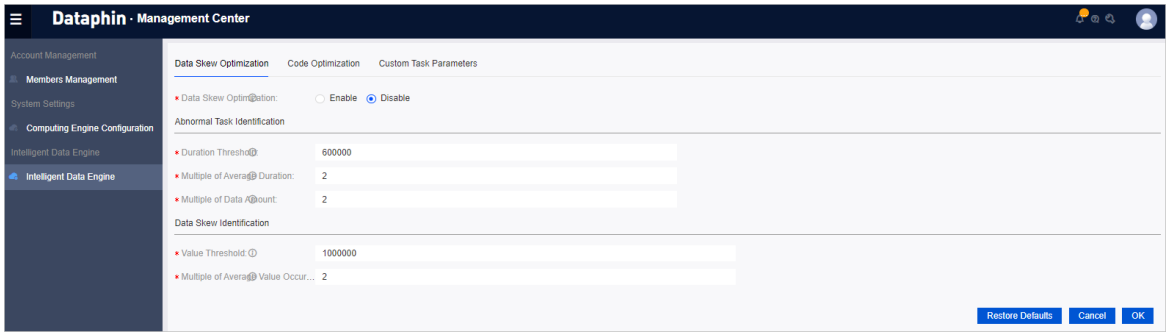
To collect MapReduce logs in real time for data cleansing, you must set YARN parameters. If you do not specify any of the following three parameters, the global switch of the data skew optimization feature is turned off by default for the intelligent data engine.

- MapReduce Log Web Service: the YARN environment variable mapreduce.jobhistory.webapp.address that is used to collect MapReduce logs in real time.
- Yarn Web Service: the YARN environment variable yarn.web-proxy.address that is used to distribute requests for accessing the YARN Resource Manager.
- Yarn Resource Manager: the YARN environment variable yarn.resourcemanager.webapp.address that is used to access the YARN Resource Manager.

Configure the data skew optimization feature

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Management Center** in the top navigation bar.

3. In the left-side navigation pane, click **Intelligent Data Engine**. The Data Skew Optimization tab appears.



4. Set relevant parameters as prompted on the Data Skew Optimization tab. The following table describes the parameters.

Parameter	Description	Example
Data Skew Optimization	Specifies whether to enable the data skew optimization feature for the intelligent data engine.	Enable
Duration Threshold	The duration threshold of an abnormal task. If the duration of a task exceeds this threshold, the intelligent data engine detects data for the task to analyze and identify anomalies. Unit: milliseconds.	600000
Multiple of Average Duration	The multiples of the average duration of all tasks. If the duration of a task is longer than the specified multiples of the average duration of all tasks, the intelligent data engine detects data for the task to analyze and identify anomalies.	2
Multiple of Data Amount	The multiples of the average data volume processed by all tasks. If the data volume processed by a task is greater than the specified multiples of the average data volume processed by all tasks, the intelligent data engine detects data for the task to analyze and identify anomalies.	2
Value Threshold	The threshold of skewed data. If a value in a dataset exceeds this threshold, the value is skewed.	1000000
Multiple of Average Value Occurrences	The multiples of the average occurrences of all values in a dataset. If the occurrences of a value in a dataset are greater than the specified multiples of the average occurrences of all values in the dataset, the value is skewed.	2

5. After the preceding parameters are set, click **OK**. In the dialog box that appears, click **OK**.

The configuration is changed.

On the **Data Skew Optimization** tab, you can also click **Restore Defaults**. In the dialog box that appears, click **OK**. The default configuration is applied.

9.4.4.2. Code optimization

This topic describes how to use the code optimization feature to optimize production code.

Introduction to the code optimization feature

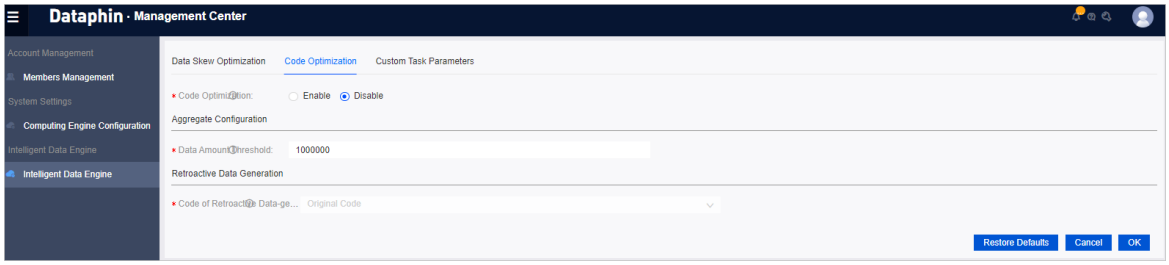
You can enable the code optimization feature to reuse the semantics of SQL statements and optimize production code. Currently, you can optimize only derived metrics. For example, the payment amount of a category on the last day can be accumulated to calculate the payment amount of the category in the last 30 days.

> **Note** The statistical dimensions of derived metrics may change.

The data in an incremental fact table for order payment is associated with products and categories. Based on this fact table, you can calculate the payment amount of each category in the last 30 days. Assume that the category of a product changes from category1 to category2 on a day of the last 30 days. Before code optimization, the payment amount is calculated based on the latest dimensions. In this case, the payment amount of the product in the last 30 days is calculated in category2. After code optimization, the payment amounts of the product in category1 and category2 in the last 30 days are still calculated in respective categories.

Configure the code optimization feature

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Management Center** in the top navigation bar.
3. In the left-side navigation pane, click **Intelligent Data Engine**. On the page that appears, click the **Code Optimization** tab.



4. On the Code Optimization tab that appears, set the following parameters:
   - **Data Amount Threshold**: the threshold of the data volume in a logical table for triggering code optimization. If the data volume of a logical table exceeds this threshold, code optimization is triggered to enable the intelligent data engine to refactor the code and optimize data computing.
   - **Code of Retroactive Data-generated Task**: the code to be run to generate retroactive data for historical tasks before the code is optimized. Code optimization is triggered based on the analysis of derived metrics and logical aggregate tables. To generate retroactive data for historical tasks before the code is optimized, you must select the code to be run. You can select the original code or optimized code. This parameter setting is not yet available.

- **Code Optimization**: specifies whether to enable the code optimization feature. You can select **Enable** or **Disable**.

5. After the preceding parameters are set, click **OK**. In the dialog box that appears, click **OK**. The configuration is changed.

   On the **Code Optimization** tab, you can also click **Restore Defaults**. In the dialog box that appears, click **OK**. The default configuration is applied.

9.4.4.3. Custom task parameters

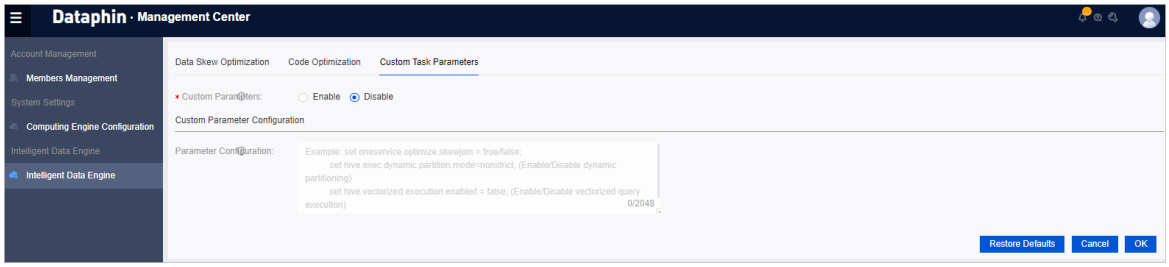
This topic describes how to set custom parameters related to the computing engine to guarantee proper resource allocation.

Introduction to the custom task parameters feature

You can set custom parameters related to the computing engine for logical table tasks to improve the efficiency of computing tasks. By setting these parameters, you can manually control the computing engine to guarantee that varied computing engines used by enterprises can adapt to Dataphin.

Configure the custom task parameters feature

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Management Center** in the top navigation bar.
3. In the left-side navigation pane, click **Intelligent Data Engine**. On the page that appears, click the **Custom Task Parameters** tab.



4. On the Custom Task Parameters tab that appears, set the following parameters:
   - **Custom Parameters**: specifies whether the custom parameters are valid. After you set custom parameters, you can apply them to the code generation rules of the intelligent data engine to control the resource allocation and operations mechanism for running tasks. For example, you can set the default memory allocated to tasks, set the default priority, and enable or disable MapJoin.
   - **Parameter Configuration**: In this field, set custom parameters based on the actual situation. The parameter settings must be consistent with the features of the computing engine configured in Dataphin. Example:

```
set odps.sql.mapper.cpu=100
set odps.sql.mapper.split.size=256
set odps.sql.reducer.cpu=100
set odps.sql.joiner.cpu=100
```

> **Note** Custom parameter settings take effect based on the priority. Task parameters enjoy the highest priority, whereas global parameters enjoy the lowest priority.

5. After the preceding parameters are set, click **OK**. In the dialog box that appears, click **OK**. The configuration is changed.

   On the **Custom Task Parameters** tab, you can also click **Restore Defaults**. In the dialog box that appears, click **OK**. The default configuration is applied.

9.5. Notifications

9.5.1. View and handle messages

This topic describes how to view and handle messages in Message Center.

View and handle unread messages

1. Log on to the Dataphin console.
2. On the Dataphin homepage, move the pointer over the icon in the top navigation bar and select **Notifications**.In the **Notifications** pane, you can view some messages on the **Messages** tab.
3. In the **Notifications** pane, click **View All** in the lower part.
4. On the page that appears, click **Unread Messages** under **Message Center** in the left-side navigation pane.
5. On the **Unread Messages** page, you can click the **Permission Management**, **Process Control**, and **Data Governance** tabs to view corresponding messages.
6. On the **All** tab, you can mark unread messages as read by using one of the following methods:
   - Find an unread message and click the icon in the **Actions** column.
   - Select multiple unread messages and click **Mark as Read** in the lower part of the page.
   - To mark all unread messages as read, click **Mark All as Read** in the lower part of the page.

View and handle all messages

1. In the left-side navigation pane, click **All Messages** under **Message Center**.
2. On the **All Messages** page, you can click the **Permission Management**, **Process Control**, and **Data Governance** tabs to view corresponding messages, either unread or read.
3. On the **All** tab, you can mark unread messages as read by using one of the following methods:
   - Find an unread message and click the icon in the **Actions** column.
   - Select multiple unread messages and click **Mark as Read** in the lower part of the page.
   - To mark all unread messages as read, click **Mark All as Read** in the lower part of the page.

9.5.2. View and handle tasks

Task Center allows you to manage all process-related tasks in Dataphin. This topic describes how to view and handle tasks in Task Center.

Context

This topic uses the permission application and approval process as an example. This process consists of the following actions:

1. Submit an application.
2. Wait for approval.
3. Approve, reject, or transfer the application, or add an approver.
4. Withdraw the application.

View and handle unprocessed tasks

1. Log on to the Dataphin console.
2. On the Dataphin homepage, move the pointer over the  icon in the top navigation bar and select **Notifications**.In the **Notifications** pane, you can view some tasks on the **Tasks** tab.
3. In the **Notifications** pane, click **View All** in the lower part.The **Unprocessed Tasks** page under **Task Center** appears.
4. On the **Unprocessed Tasks** page, you can view the tasks that need to be handled.You can find unprocessed tasks by using one of the following methods:
   - Click **Permission Approval** next to **Task Type** to filter the tasks that require you to approve permission applications.
   - Enter a keyword of the task name in the search box to search for corresponding unprocessed tasks.
5. On the **Unprocessed Tasks** page, click the name of a task in the **Task Name** column.
6. Handle the unprocessed task.

i. On the page that appears, you can view details of the task. In the **Task Processing** section, set the parameters as required.

Parameter	Description
Approval Result	The approval result of the task. Valid values: ■ **Approve**: approves the application in the task and grants permissions to the applicant. ■ **Reject**: rejects the application in the task and requests the applicant to apply again or modify the application content. ■ **Transfer To**: transfers the task to another person. You can select a person to whom the task is transferred. After you transfer the task, the selected person is responsible for processing the application in the task. ■ **Add Approver**: adds an approver. You can select a person as a new approver. ■ After the new approver approves the application in the task, the task will be transferred back to you. ■ If the new approver rejects the application in the task, the process ends.
Comments	The comments of the approval.

ii. After you set the preceding parameters, click **Submit**.

View processed tasks

1. In the left-side navigation pane, click **Processed Tasks** under **Task Center**.
2. On the **Processed Tasks** page, you can view the processed tasks.You can find processed tasks by using one of the following methods:
   - Set the **Task Type** and **Status** parameters to filter processed tasks.
   - Enter a keyword of the task name in the search box to search for corresponding processed tasks.
3. On the Processed Tasks page, click the name of a task in the **Task Name** column to go to the details page of the task.
4. After you view the task details, click **Cancel** in the lower part of the page.

View tasks that you initiated

1. In the left-side navigation pane, click **Initiated Tasks** under **Task Center**.
2. On the **Initiated Tasks** page, you can view the tasks you initiated.You can find initiated tasks by using one of the following methods:
   - Set the **Task Type** and **Status** parameters to filter corresponding tasks you initiated.
   - Enter a keyword of the task name in the search box to search for corresponding tasks you initiated.
3. On the Initiated Tasks page, click the name of a task in the **Task Name** column to go to the

details page of the task.

4. After you view the task details, click **Cancel** in the lower part of the page.

9.6. Alert Center

9.6.1. Overview

Alert Center is a monitoring and alerting module in Dataphin. It displays alert events, push records, and shift schedules of the Data Quality, Stream Processing, and API Service modules.

Background information

As Dataphin supports more and more features in data collection, data warehouse building, data management, and data application, it also needs to support more complex alert scenarios.

In earlier versions, Dataphin has been able to monitor errors, latency, and timeout exceptions of scheduling tasks and generate alerts when specific conditions are met. However, you can select only a limited number of alert causes to configure alert rules. In this situation, you need a monitoring and alerting center that provides enhanced monitoring capabilities and more monitoring scenarios.

Features

Alert Center provides the following pages:

- **Events**: This page displays alert events of the Stream Processing, Data Quality, and API Service modules.
- **Push Records**: This page displays records about alert messages that Alert Center pushes.
- **Shift Schedules**: This page displays shift schedules that are arranged for alert recipients who need to handle alerts.

Log on to the Dataphin console. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.

9.6.2. Terms

This topic describes terms involved in Alert Center, including alert event, push record, shift schedule, and push method.

Term	Description
Alert event	An event generated based on the specific alert object and alert rule.
Push record	A record generated after Alert Center pushes an alert message to the message subscribers.
Shift schedule	A schedule arranged for alert recipients who need to handle alerts.
Push method	A method that Alert Center uses to push an alert message to the message subscribers.

Term	Description
Do-Not-Disturb	A feature that allows you to specify a period during which Alert Center does not push any alert messages to you. For example, if Alert Center repeatedly pushes alert messages of the same alert event to you, you can specify a Do-Not-Disturb period for the alert event.
Service	A module where an alert event was triggered. Currently, alert events may be triggered in Data Quality, Stream Processing, and API Service.

9.6.3. Events

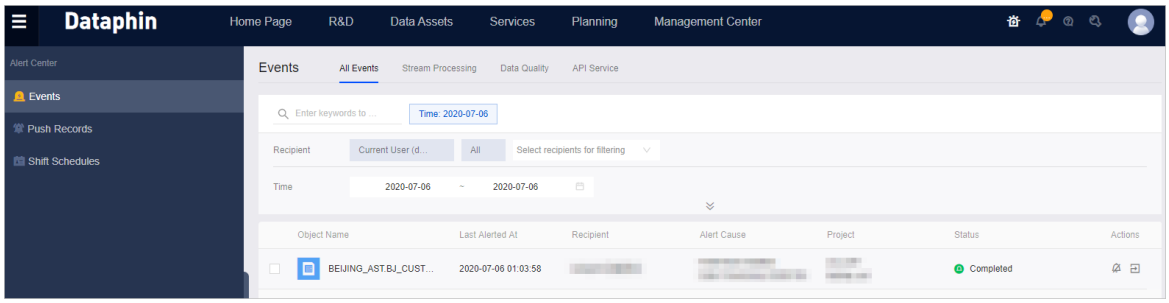
The Events page displays alert events of the Stream Processing, Data Quality, and API Service modules. This topic describes how to view and handle alert events.

Go to the Events page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to Alert Center.

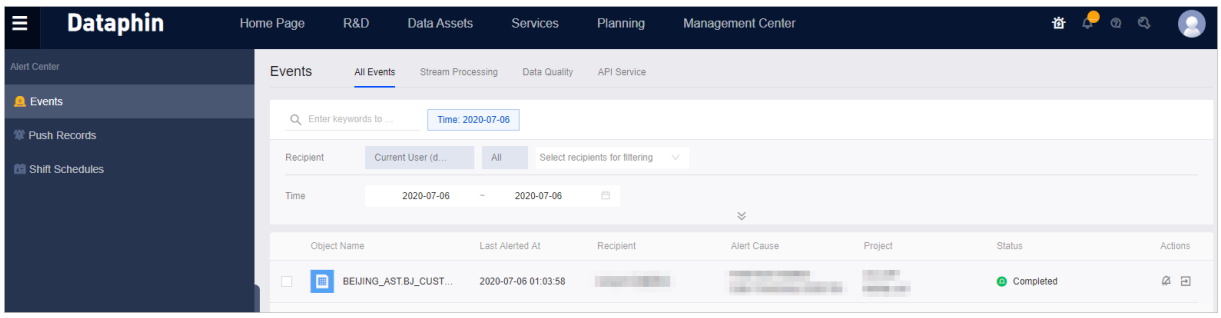
   On the **Events** page, you can click the **Stream Processing**, **Data Quality**, or **API Service** tab to view alert events of the corresponding module.

   To view all alert events of the **Stream Processing**, **Data Quality**, and **API Service** modules, click the **All Events** tab. This topic describes alert events on the **All Events** tab.



View alert events

On the **All Events** tab, you can set the **Recipient**, **Time**, **Status**, **Business Unit**, and **Project** parameters to filter alert events. You can also enter a keyword in the search box to search for alert events.



The **All Events** tab displays the **Object Name**, **Last Alerted At**, **Recipient**, **Alert Cause**, **Project**, and **Status** parameters of each alert event. The **Actions** column of an alert event lists the operations that you can perform on the alert event.

Parameter	Description
Object Name	The name of the object that triggered the alert event. Alert events of each module are named based on the following conventions: • **Stream Processing**: Each alert event is named after the corresponding stream processing task that is published to the production environment, for example, `flink_merge`. • **API Service**: Each alert event is named after the corresponding API operation, for example, `user_info_service`. • **Data Quality**: ○ An alert event that is triggered by a physical table is named in the `{Project name}.{Table name}` format, for example, `v27_ast.ods_user_info`. ○ An alert event that is triggered by a logical table is named in the `{Business unit name}.{Table name}` format, for example, `LD_demo.dws_all`.
Last Alerted At	The time when Alert Center pushed the last alert message for the alert event.
Recipient	The recipient to whom Alert Center pushed alert messages for the alert event. You can configure multiple recipients for an alert event, including: • The owner of the object that triggers the alert event. • Custom users. • Users in a shift schedule.

Parameter	Description
Alert Cause	The cause of the alert event. The cause of an alert event varies based on the module where the alert event may be triggered. • In **Data Quality**, an alert event may be triggered due to the following causes: **Table Rule Violation**, **Field Rule Violation**, and **Custom Rule Violation**. • In **API Service**, an alert event may be triggered due to the following causes: **Unexpected Average Response Time**, **Unexpected Number of Calls**, **Unexpected Error Rate**, and **Unexpected Percentage of Offline Tasks**. • In **Stream Processing**, an alert event may be triggered due to the following causes: **Excessively Long Delay**, **Exceeds Maximum TPS**, **Exceeds Specified Failure Frequency**, and **Exceeds Specified Data Retention Period**.
Project	The project to which the alert event belongs.
Status	The status of the alert event. Valid values: • **Completed**: The alerting process is completed for the alert event. • **Alerting**: The frequent alerting mode is set for the alert event, and the alerting start time has arrived but the end time has not. • **Do-Not-Disturb**: You can change the status of ongoing and completed alert events to **Do-Not-Disturb**. • **Alerting (Silence Period)**: Alert Center does not push alert messages for the alert event because the alert time does not arrive. Assume that API Service requires Alert Center to push alert messages for each alert event at an interval of 5 minutes. After an alert message is pushed, Alert Center does not push alert messages for the same alert event within 5 minutes even if metrics that trigger the alert event are collected every second.
Actions	The operations that you can perform on the alert event. • To handle an alert event, find the alert event and click the  icon in the **Actions** column. For more information, see Handle alert events. • To change the status of an alert event to **Do-Not-Disturb**, find the alert event and click the icon in the **Actions** column. For more information, see Change the status of alert events to Do-Not-Disturb.

View details of an alert event

On the **All Events** tab, click a name in the **Object Name** column. The **Event Details** pane appears.

GUI element	Description
Do-Not-Disturb	Allows you to change the status of the alert event to **Do-Not-Disturb**. For more information, see Change the status of alert events to Do-Not-Disturb.

GUI element	Description
Handle Alert	Allows you to handle the alert event. For more information, see Handle alert events.
Object Type	The type of the object that triggered the alert event. Valid values of the **Object Type** parameter of an alert event vary based on the module where the alert event may be triggered. • In **Data Quality**, valid values are **Logical Table** and **Physical Table**. • In **Stream Processing**, the value is fixed to **Task**. • In **API Service**, the value is fixed to **API**.
Service	The module where the alert event was triggered. Valid values: • **Data Quality** • **Stream Processing** • **API Service**
Alert Cause	The cause of the alert event. For more information, see View alert events.
Push Method	The method that Alert Center used to push alert messages for the alert event. Valid values: • **Cellphone** • **SMS** • **Email** • **DingTalk**
Alert Frequency	The frequency at which Alert Center pushed alert messages for the alert event.
First Alerted At	The time when Alert Center pushed the first alert message for the alert event.
Last Alerted At	The time when Alert Center pushed the last alert message for the alert event.
Alert Recipients	The recipients to whom Alert Center pushed alert messages for the alert event. For more information, see View alert events.
Links	The link to the module where the alert event was triggered and the link to the corresponding operational logs. The links that are provided for an alert event vary based on the module where the alert event may be triggered. • **Data Quality**: The **Go to Quality Check Rules** and **Go to Scheduling Task Instance Log** links are provided. • **Stream Processing**: The **Go to Stream Processing Task Monitoring Rules** link is provided. • **API Service**: The **Go to API Service Monitoring Rules** link is provided.

Handle alert events

1. Go to the module where an alert event was triggered by using one of the following

methods:

- On the **All Events** tab, find the alert event and click the icon in the **Actions** column.
- On the All Events tab, click the name of the object that triggered the alert event. In the **Event Details** pane, click **Handle Alert**.

2. In the module where the alert event was triggered, handle the alert event based on the alert cause.

Change the status of alert events to Do-Not-Disturb

If alert events are in the **Alerting** or **Completed** state, you can change the status to **Do-Not-Disturb**.

1. On the **All Events** tab, open the **Alert Do-Not-Disturb Settings** dialog box by using one of the following methods:
   - Find an alert event and click the icon in the **Actions** column.
   - Select multiple alert events in the **Alerting** or Completed state and click **Do-Not-Disturb** in the lower part of the page.
2. In the **Alert Do-Not-Disturb Settings** dialog box, set the **Duration** parameter to **All Day** and specify the **Specified Duration** parameter as required.
3. Click **OK**.

   The status of one or more alert events changes from **Alerting** or **Completed** to **Do-Not-Disturb**.

9.6.4. Push records

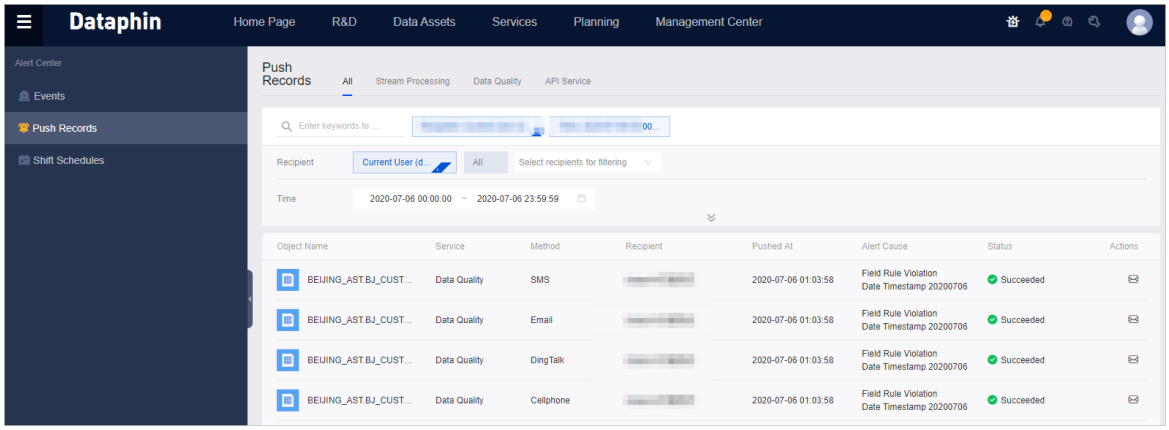
The Push Records page displays records about alert messages that Alert Center pushes. This topic describes how to view push records and the content of alert messages that are pushed.

Go to the Push Records page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In **Alert Center**, click **Push Records** in the left-side navigation pane.

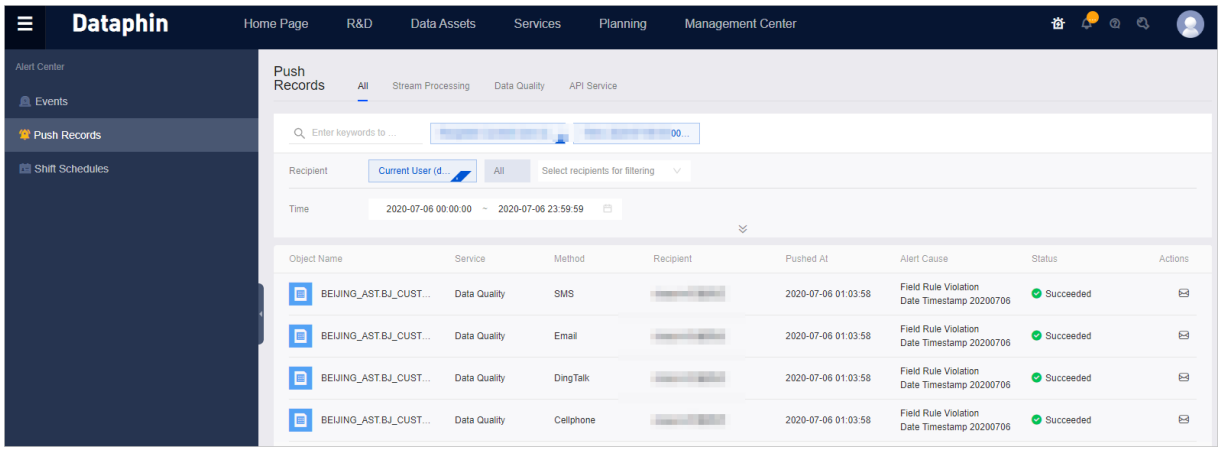
   On the **Push Records** page, click the **Stream Processing**, **Data Quality**, or **API Service** tab to view the push records of the corresponding module.

To view all push records of the **Stream Processing**, **Data Quality**, and **API Service** modules, click the **All** tab. This topic describes push records on the **All** tab.



View push records

On the **All** tab, you can set the **Recipient**, **Time**, **Status**, and **Method** parameters to filter push records. You can also enter a keyword in the search box to search for push records.



The **All** tab displays the **Object Name**, **Service**, **Method**, **Recipient**, **Pushed At**, **Alert Cause**, and **Status** parameters of each push record. The **Actions** column of a push record lists the operations that you can perform on the push record.

Parameter	Description
Object Name	The name of the object that triggered the alert event.
Service	The module where the alert event was triggered. Valid values: • **Data Quality** • **Stream Processing** • **API Service**

Parameter	Description
Method	The method that Alert Center used to push the alert message to the specific recipients. Valid values: • Cellphone • SMS • Email • DingTalk
Recipient	The recipients of the alert message.
Pushed At	The time when the alert message was pushed.
Alert Cause	The cause of the alert event for which the alert message was pushed. The cause of an alert event varies based on the module where the alert event may be triggered. • In **Data Quality**, an alert event may be triggered due to the following causes: **Table Rule Violation**, **Field Rule Violation**, and **Custom Rule Violation**. • In **API Service**, an alert event may be triggered due to the following causes: **Unexpected Average Response Time**, **Unexpected Number of Calls**, **Unexpected Error Rate**, and **Unexpected Percentage of Offline Tasks**. • In **Stream Processing**, an alert event may be triggered due to the following causes: **Excessively Long Delay**, **Exceeds Specified Data Retention Period**, **Exceeds Specified Failure Frequency**, and **Exceeds Maximum TPS**.
Status	The pushing status of the alert message. Valid values: • Sending • Succeeded • Failed
Actions	To view the content of an alert message that was pushed, find the push record and click the icon in the **Actions** column.

View the content of an alert message that was pushed

On the **Push Records** page, find a push record and click the icon in the **Actions** column.

9.6.5. Shift schedules

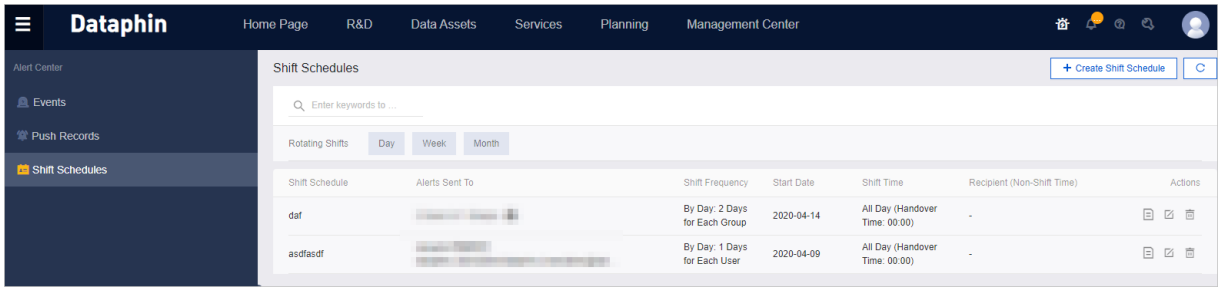
9.6.5.1. Overview

The Shift Schedules page displays shift schedules that are arranged for alert recipients who need to handle alerts.

To go to the **Shift Schedules** page, perform the following steps:

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In the left-side navigation pane, click **Shift Schedules**.

On the **Shift Schedules** page, you can set the **Rotating Shifts** parameter to filter shift schedules. You can also enter a keyword in the search box to search for shift schedules.



The **Shift Schedules** page displays the **Shift Schedule**, **Alerts Sent To**, **Shift Frequency**, **Start Date**, **Shift Time**, and **Recipient (Non-Shift Time)** parameters of each shift schedule. The **Actions** column of a shift schedule lists the operations that you can perform on the shift schedule.

Parameter	Description
Shift Schedule	The name of the shift schedule.
Alerts Sent To	The recipients to whom Alert Center pushes alert messages during shifts that are arranged in the shift schedule. • If you set the Alerts Sent To parameter to **Users** when you create a shift schedule, the Alerts Sent To column of the shift schedule displays the specific user accounts. This column displays a maximum of 64 characters for each value. If a value contains more than 64 characters, excess characters are displayed as an ellipsis (...). • If you set the Alerts Sent To parameter to **Groups** when you create a shift schedule, the Alerts Sent To column of the shift schedule displays the number of groups and total number of group members.You can move the pointer over the icon to view group details.
Shift Frequency	The rotating frequency of shifts that are arranged in the shift schedule. For more information, see Create a shift schedule.
Start Date	The start date of shifts that are arranged in the shift schedule. For more information, see Create a shift schedule.
Shift Time	The duration of a shift. For more information, see Create a shift schedule.
Recipient (Non-Shift Time)	The recipient to whom Alert Center pushes alert messages in the non-shift duration of specific alert recipients. For example, the recipient can be the object owner or a specified user.

Parameter	Description
Actions	The operations that you can perform on the shift schedule, including: • View details of the shift schedule. For more information, see View details of a shift schedule. • Modify the shift schedule. For more information, see Modify a shift schedule. • Delete the shift schedule. For more information, see Delete a shift schedule.

9.6.5.2. Create a shift schedule

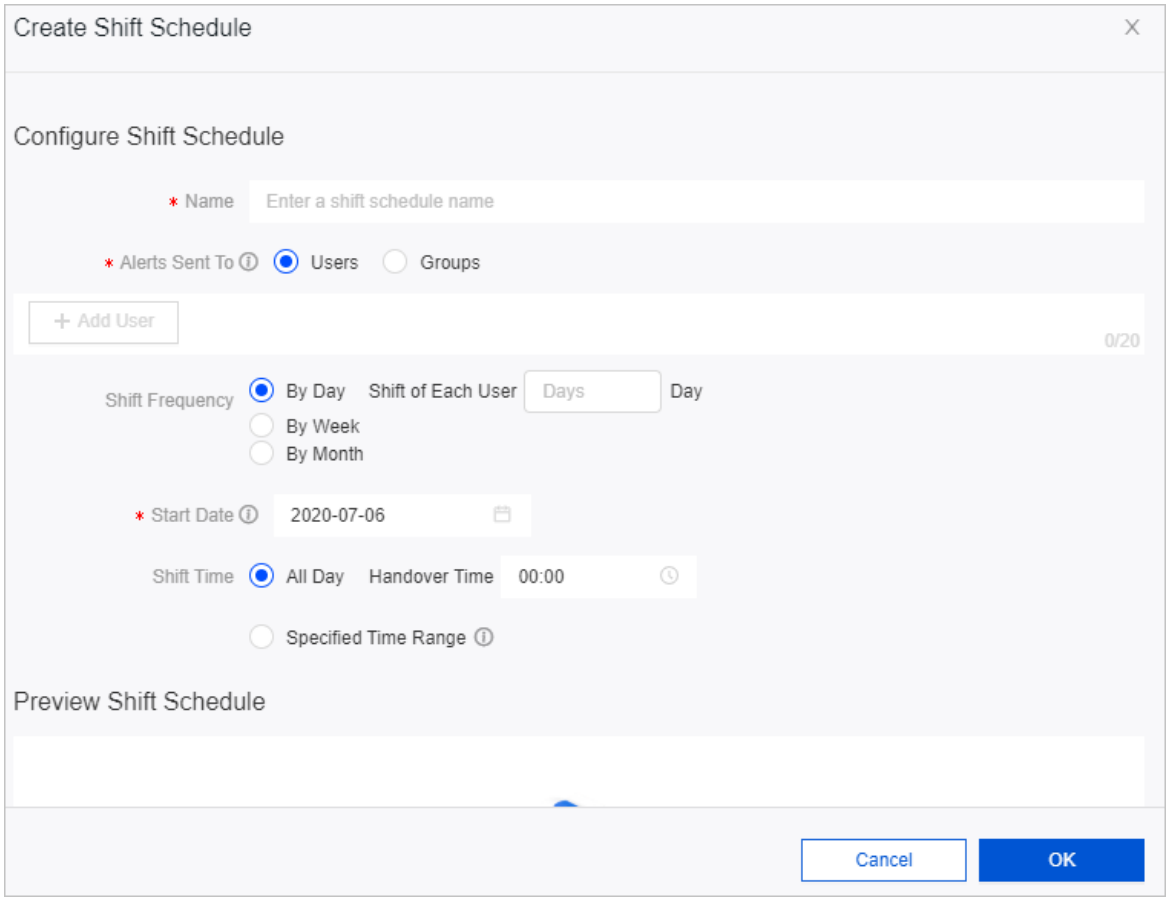
The Shift Schedules page displays shift schedules that are arranged for alert recipients who need to handle alerts. This topic describes how to create a shift schedule.

Context

Only the super administrator and business unit administrators can create a shift schedule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In the left-side navigation pane, click **Shift Schedules**.
4. On the **Shift Schedules** page, click **Create Shift Schedule** in the upper-right corner.
5. In the **Create Shift Schedule** dialog box, set the parameters as required.



Section	Parameter	Description
	Name	The name of the shift schedule.
	Alerts Sent To	The alert recipients for the shift schedule. Valid values: **Users** and **Groups**. ◦ If you select **Users**, click **Add User** to add one or more users as alert recipients. **Note** If the icon appears next to a user to be added as an alert recipient, you must add contact information for the user. For more information, see Manage members. ◦ If you select **Groups**, click **Add User** next to **Group 1** and add alert recipients to the first group. If you want alert recipients in multiple groups to take turns, click **Add Group** to add more groups and add alert recipients to the groups. If an existing group is no longer needed, move the pointer over the group name and click the icon to delete the group.

Section	Parameter	Description
Configure Shift Schedule	Shift Frequency	The rotating frequency of shifts that are arranged in the shift schedule. Valid values: ○ **By Day**: If you select this option, the value in the Shift of Each User or Shift of Each Group field next to this option cannot exceed 366, in units of days. ○ **By Week**: If you select this option, the value in the Shift of Each User or Shift of Each Group field next to this option cannot exceed 53, in units of weeks. ○ **By Month**: If you select this option, the value in the Shift of Each User or Shift of Each Group field next to this option cannot exceed 12, in units of months.
	Start Date	The start date of shifts that are arranged in the shift schedule.
	Shift Time	The duration of a shift. Valid values: **All Day** and **Specified Time Range**. ○ If you select **All Day**, specify the **Handover Time** parameter. ○ If you select **Specified Time Range**, specify a time range and the **Recipient (Non-Shift Time)** parameter. Valid values of the **Recipient (Non-Shift Time)** parameter are **Object Owner** and **Specified User**.
Preview Shift Schedule	Preview	You can click **Preview** to view the configured shift schedule. After you update the shift schedule, you can click **Refresh Preview** to update the preview data of the shift schedule.

6. Click **OK**.

9.6.5.3. View details of a shift schedule

The Shift Schedules page displays shift schedules that are arranged for alert recipients who need to handle alerts. This topic describes how to view details of a shift schedule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In the left-side navigation pane, click **Shift Schedules**.
4. On the **Shift Schedules** page, find the shift schedule whose details you want to view and click the icon in the **Actions** column.

9.6.5.4. Modify a shift schedule

The Shift Schedules page displays shift schedules that are arranged for alert recipients who need to handle alerts. This topic describes how to modify a shift schedule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In the left-side navigation pane, click **Shift Schedules**.
4. On the Shift Schedules page, find the shift schedule that you want to modify and click the icon in the **Actions** column.
5. In the **Change Shift Schedule** dialog box, modify the parameters as needed. For more information, see Create a shift schedule.
6. After you modify the parameters, click **OK**.

9.6.5.5. Delete a shift schedule

The Shift Schedules page displays shift schedules that are arranged for alert recipients who need to handle alerts. This topic describes how to delete a shift schedule.

Context

Limits:

- You cannot delete a shift schedule that is being referenced by the Stream Processing, Data Quality, or API Service module.
- Only the super administrator and business unit administrators can delete a shift schedule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click the icon in the upper-right corner to go to **Alert Center**.
3. In the left-side navigation pane, click **Shift Schedules**.
4. On the **Shift Schedules** page, find the shift schedule that you want to delete and click the icon in the **Actions** column.
5. In the **Tip** message, click **OK**.

9.7. Data warehouse planning

9.7.1. Overview

Data warehouse planning is a key step in architectural design (during data production). To complete this step, you need to define logical space divisions based on your business characteristics. The logical space divisions include business units, data domains, namespaces, and global objects. Then, create projects based on your development cooperation and management mode. When creating a project, configure project and member settings, and register the required underlying data sources (physical databases).

9.7.2. Business unit management

9.7.2.1. Create business units

As an important part of logical spaces, a business unit defines a data warehouse namespace in Dataphin based on business characteristics. A business unit can contain multiple projects. This topic describes how to create business units in Dev-Prod mode or a business unit in Basic mode.

Prerequisites

Dataphin members are added. For more information, see Manage members.

Context

Only the Apsara Stack tenant account that is assigned the super administrator role can create business units.

Dataphin allows you to create business units in Dev-Prod or Basic mode. The following table describes the two modes.

Mode	Description
Dev-Prod mode	Generates a business unit in Dev mode as the development environment and a business unit in Prod mode as the production environment. The two business units are isolated. This allows you to guarantee data security in the production environment and manage the data production process and production data in a controllable way. We recommend that you select this mode if you are concerned more about management. This mode is applicable to the scenario where you have many developers who have clear responsibilities and you have a large budget for computing and storage.
Basic mode	Generates a business unit in Basic mode as an independent and flexible production environment. In this mode, the data production process is stable and easy-to-use and the production data is controllable. We recommend that you select this mode if you are concerned more about data development efficiency rather than management. This mode is applicable to the scenario where the responsibilities of developers are not clear and you have a limited budget for computing and storage.

Create business units in Dev-Prod mode

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, click **Create Business Unit** in the upper-right corner.
5. In the **Create Business Unit** dialog box, select Dev-Prod Mode, click Next, and then set the parameters as required.

Parameter	Description
Common Name	The common name of the business units. The common name can be up to 64 characters in length and can contain letters, digits, and underscores (_).
Common Display Name	The common display name of the business units. The common display name can be up to 64 characters in length and can contain letters, digits, underscores (_), and hyphens (-). Note By default, the name of the business unit in Dev mode is suffixed with _dev.
Description	The description of the business units.
Icon	The icon that indicates the category of the business units. Valid values: ○  : E-Commerce ○  : Finance ○  : Cloud Computing ○  : Advertising and Marketing ○  : Logistics ○  : Entertainment ○  : Travel ○  : Health ○  : Socialization and Communications ○  : Food & Drink ○  : Education ○  : Environment

Parameter	Description
Business Unit Administrators	The administrator of the business units. You can specify multiple administrators.

6. Click **OK**.

Create a business unit in Basic mode

1. On the **Business Units** page, click **Create Business Unit** in the upper-right corner.
2. In the **Create Business Unit** dialog box, select Basic Mode, click **Next**, and then set the parameters as required.

Parameter	Description
Name	The name of the business unit. The name can be up to 64 characters in length and can contain letters, digits, and underscores (_).
Display Name	The display name of the business unit. The display name can be up to 64 characters in length and can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the business unit.

Parameter	Description
Icon	The icon that indicates the category of the business unit. Valid values: ○  : E-Commerce ○  : Finance ○  : Cloud Computing ○  : Advertising and Marketing ○  : Logistics ○  : Entertainment ○  : Travel ○  : Health ○  : Socialization and Communications ○  : Food & Drink ○  : Education ○  : Environment
Business Unit Administrators	The administrator of the business unit. You can specify multiple administrators.

3. Click **OK**.

9.7.2.2. Modify a business unit

As an important part of logical spaces, a business unit defines a data warehouse namespace in Dataphin based on business characteristics. A business unit can contain multiple projects. This topic describes how to modify a business unit.

Prerequisites

Business units are created. For more information, see Create business units.

Context

Only the Apsara Stack tenant account that is assigned the super administrator role can modify business units.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, move the pointer over the business unit that you want to modify and click the icon.
5. In the **Change Business Unit** dialog box, change the parameter values as needed. For more information, see Create business units.
6. Click **OK**.

9.7.2.3. Modify business unit parameters

Business unit parameters include the time-based partitioning parameter and the data timestamp parameter. This topic describes how to modify business unit parameters.

Prerequisites

Business units are created. For more information, see Create business units.

Context

Only the Apsara Stack tenant account that is assigned the super administrator role can modify business unit parameters.

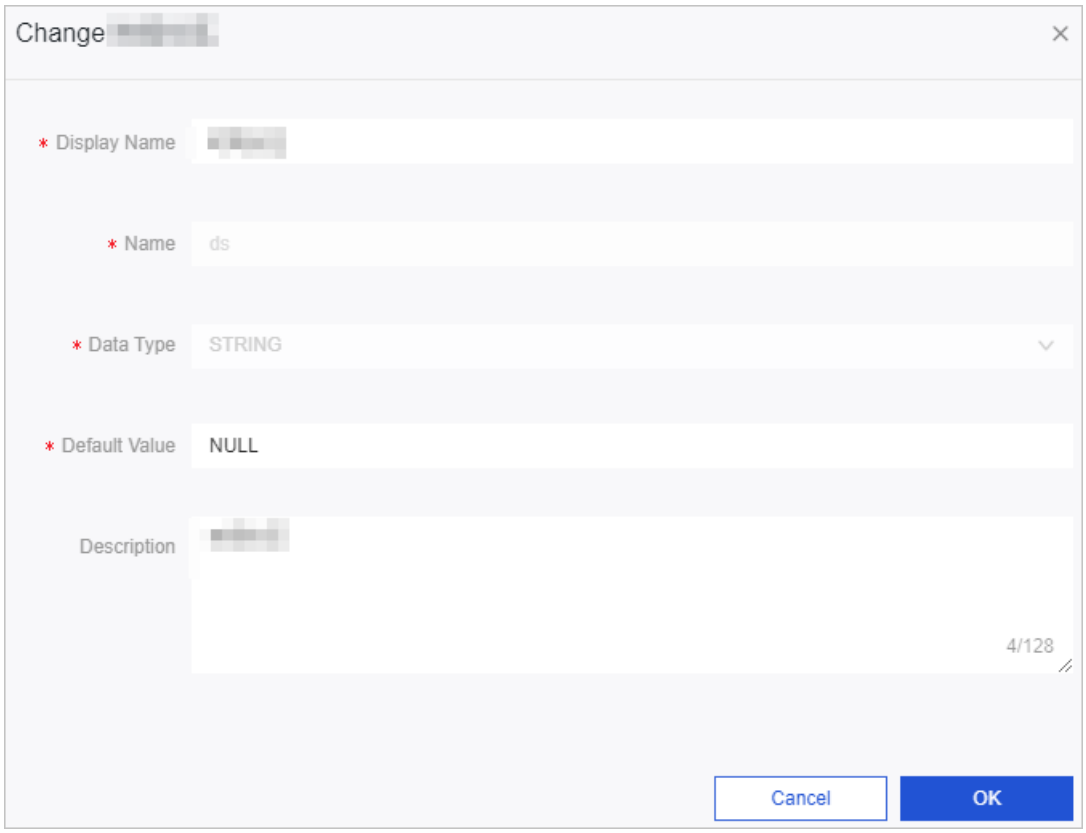
Business unit parameters function as common parameters of a business unit, such as the data type and default value. The following business unit parameters are supported:

- Time-based Partitioning Field: specifies the name, data type, default value, and description of the time-based partitioning parameter.
- Data Timestamp: specifies the name and format of the data timestamp parameter. By default, the data timestamp parameter is named bizdate.

Modify the time-based partitioning parameter

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.

4. On the **Business Units** page, click the business unit for which you want to modify business unit parameters.
5. In the **Business Unit Parameters** section, click the icon next to **Time-based Partitioning Field**.
6. In the **Change Time Partition** dialog box, set the parameters as required.



Parameter	Description
Display Name	The display name of the time-based partitioning parameter. The display name can contain letters, digits, underscores (_), and hyphens (-).
Name	The name of the time-based partitioning parameter. Default value: **ds**. The name can contain letters, digits, and underscores (_).
Date Type	The data type of the time-based partitioning parameter. Valid values: ○ **STRING** ○ **BIGINT** ○ **DOUBLE** ○ **DATETIME**
Default Value	The default value of the time-based partitioning parameter. Enter NULL if you do not specify a default value.
Description	The description of the time-based partitioning parameter.

7. Click **OK**.

Modify the data timestamp parameter

1. On the **Business Units** page, click the business unit for which you want to modify business unit parameters.
2. In the **Business Unit Parameters** section, click the icon next to **Data Timestamp**.
3. In the **Change Data Timestamp** dialog box, click the icon next to **Parameter Value** and select a format for the data timestamp from the drop-down list.
4. Click **OK**.

9.7.2.4. Data domain management

9.7.2.4.1. Create a data domain

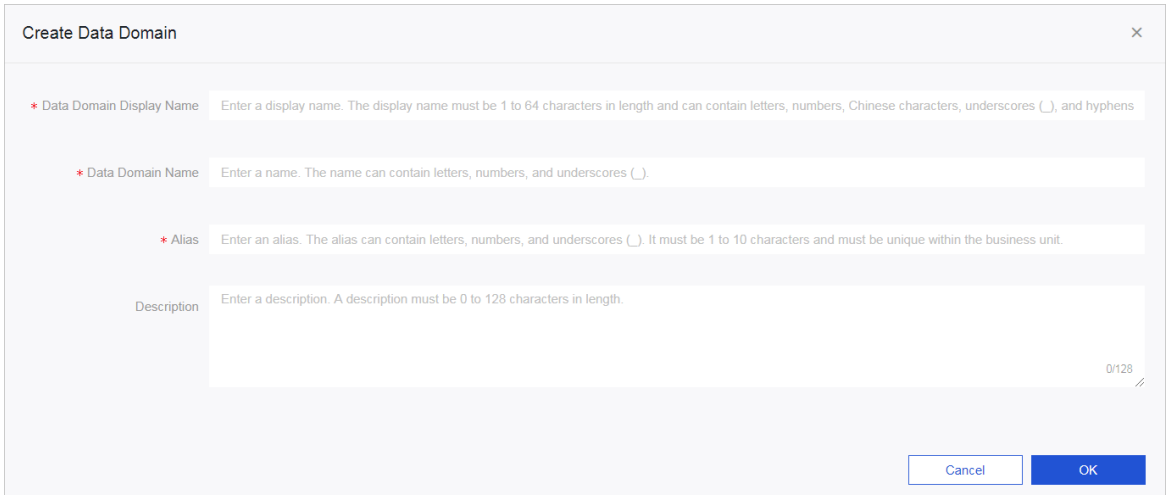
A data domain is used to categorize business concepts in a business unit, such as the commodity domain, transaction domain, and membership domain. This topic describes how to create a data domain.

Context

Only the super administrator and project administrators can create data domains.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, click the business unit for which you want to create a data domain.
5. In the Data Domain section, Click **Create Data Domain**.
6. In the **Create Data Domain** dialog box, set the parameters as required.



Parameter	Description
Data Domain Display Name	The display name of the data domain.
Data Domain Name	The name of the data domain.
Alias	The alias of the data domain. The alias cannot be the same as the name of the data domain.
Description	The description of the data domain.

7. Click **OK**.

9.7.2.4.2. Modify a data domain

A data domain is used to categorize business concepts in a business unit, such as the commodity domain, transaction domain, and membership domain. This topic describes how to modify a data domain.

Context

If a data domain is referenced, changing its name may result in errors in dependencies. In this case, we recommend that you do not modify the data domain name.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, click the business unit for which you want to modify a data domain.
5. In the **Data Domain** section, find the data domain that you want to modify and click the icon in the **Actions** column.
6. In the **Change Data Domain** dialog box, change the parameter values as needed. For more information, see Create a data domain.
7. Click **OK**.

9.7.2.4.3. Delete a data domain

A data domain is used to categorize business concepts in a business unit, such as the commodity domain, transaction domain, and membership domain. This topic describes how to delete a data domain.

Context

You cannot delete a data domain if it is referenced.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, click the business unit for which you want to delete a data domain.
5. In the **Data Domain** section, find the data domain that you want to delete and click the icon in the **Actions** column.
6. In the message that appears, click **OK**.

9.7.2.5. Delete a business unit

As an important part of logical spaces, a business unit defines a data warehouse namespace in Dataphin based on business characteristics. A business unit can contain multiple projects. This topic describes how to delete a business unit.

Prerequisites

Business units are created. For more information, see Create business units.

Context

Limits:

- Only the Apsara Stack tenant account that is assigned the super administrator role can delete business units.
- To delete business units in Dev-Prod mode, make sure that the projects in the business units contain no tasks or objects.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Business Units** in the left-side navigation pane.
4. On the **Business Units** page, move the pointer over the business unit that you want to delete and click the icon.
5. In the **Delete Business Unit** message, click **OK**.

9.7.3. Statistical period management

9.7.3.1. Create a statistical period

Dataphin supports only the common business logic of statistical periods. A statistical period indicates the time range for statistics, for example, the last seven days or the last 30 days. This topic describes how to create a statistical period.

Prerequisites

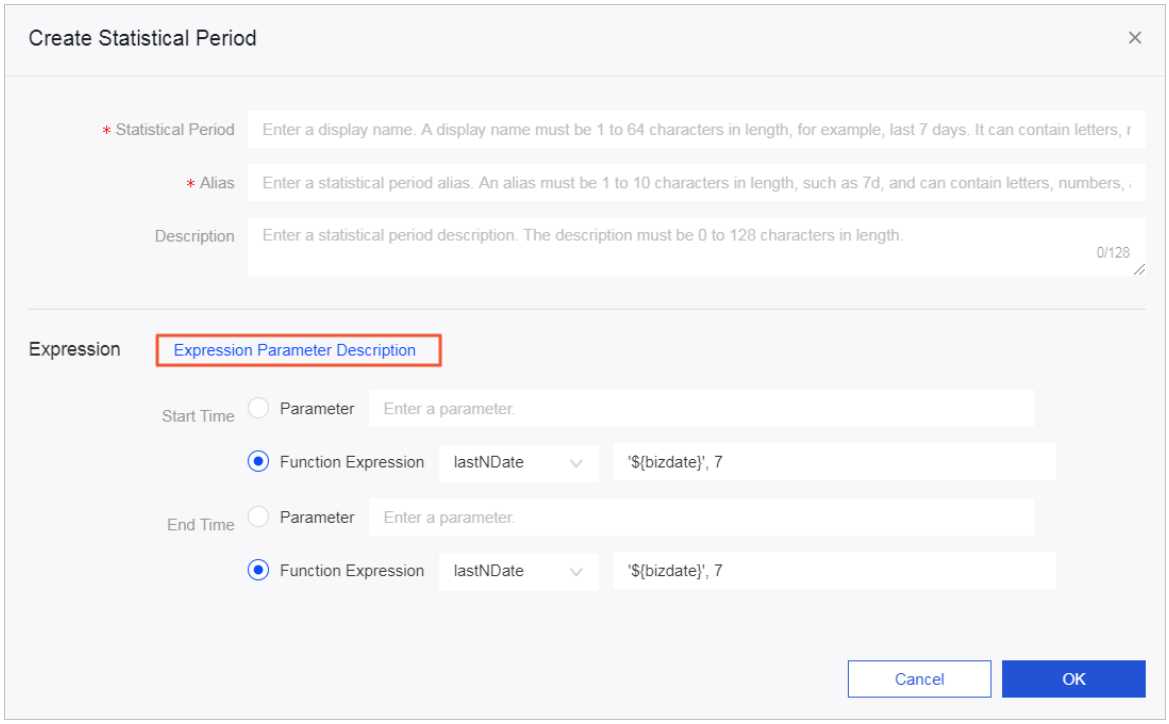
The computing engine type is set. For more information, see Set the computing engine type. After the computing engine type is set, Dataphin automatically initializes the built-in statistical periods.

Context

Only the super administrator and project administrators can create statistical periods.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Common Business Logic** in the left-side navigation pane.
4. On the **Common Business Logic** page, click **Create Statistical Period** in the upper-right corner.
5. In the **Create Statistical Period** dialog box, set the parameters as required.



Parameter	Description
Statistical Period	The display name of the statistical period.
Alias	The alias of the statistical period.
Description	The description of the statistical period.
Expression	The function expressions of the statistical period. You can click **Expression Parameter Description** next to **Expression** to view functions that Dataphin provides.
Start Time	The expression of the start time of the statistical period. You can set the expression of the start time by using one of the following methods: ○ Select **Parameter** and enter a parameter. ○ Select **Function Expression**, click the icon, and then select a function expression.
End Time	The expression of the end time of the statistical period. You can set the expression of the end time by using one of the following methods: ○ Select **Parameter** and enter a parameter. ○ Select **Function Expression**, click the icon, and then select a function expression.

Note You cannot create a duplicate statistical period.

6. Click **OK**.

9.7.3.2. Modify a statistical period

A statistical period indicates the time range for statistics, for example, the last seven days or the last 30 days. This topic describes how to modify a statistical period.

Prerequisites

Statistical periods are created. For more information, see Create a statistical period.

Context

Limits:

- The super administrator and project administrators can modify statistical periods.
- A project administrator cannot modify statistical periods that are created by other administrators. Members who are not administrators can modify only the statistical periods that are created by themselves.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Common Business Logic** in the left-side navigation pane.
4. On the **Common Business Logic** page, find the statistical period that you want to modify and click the icon in the **Actions** column.
5. In the **Change Statistical Period** dialog box, change the parameter values as needed. For more information, see Create a statistical period.
6. Click **OK**.

9.7.3.3. Delete a statistical period

A statistical period indicates the time range for statistics, for example, the last seven days or the last 30 days. This topic describes how to delete a statistical period.

Prerequisites

Statistical periods are created. For more information, see Create a statistical period.

Context

Limits:

- The super administrator and project administrators can delete statistical periods.
- A project administrator cannot delete statistical periods that are created by other administrators. Members who are not administrators can delete only the statistical periods that are created by themselves.
- If a published derived metric references a statistical period, such as the last 150 days, the statistical period cannot be deleted.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Common Business Logic** in the left-side navigation pane.
4. On the **Common Business Logic** page, find the statistical period that you want to delete and click the icon in the **Actions** column.
5. In the message that appears, click **OK**.

9.7.4. Project management

9.7.4.1. Overview

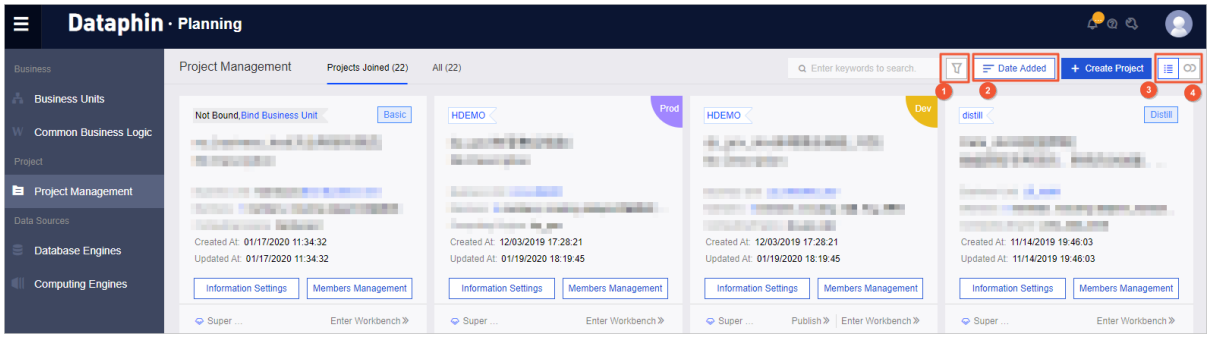
A project is used to isolate physical resources and developers during Data Mid-End construction. A business unit can contain multiple projects. Each Dataphin member can join multiple projects.

> **Note** Before you manage a project, make sure that the computing engine type of Dataphin is set on the Computing Engine Configuration page in Management Center. For more information, see Set the computing engine type.

On the **Project Management** page, you can find the projects that you join on the **Projects Joined** tab and all projects on the **All** tab. You can perform the following operations on the projects that you join:

- Configure project information, manage members, go to the **Publish** page, and go to the **Develop** page.
- In the **Information Settings** dialog box, configure the batch processing computing engine and stream processing computing engine for a project. For more information, see Create a batch processing computing engine and Create a stream processing computing engine. Dataphin supports stream processing and batch processing. You can develop both stream processing tasks and batch processing tasks in the same project. This ensures consistent experience in development and improves the development efficiency.

You can also filter and sort projects and change the display mode of projects on the **Project Management** page, as described in the following table.



No.	Description
1	Click the icon. In the **More Filters** dialog box, set the **Business Units** and **Project Type** parameters. Projects that meet the filter conditions appear.
2	Click the icon and select **Date Added**, **Display Name**, or **Name** to display projects based on the dates when the projects are added in descending order or based on the display names or names in reverse alphabetical order.
3	Click the icon. The projects are listed.
4	Click the icon. The projects are categorized based on the project mode and projects in Dev-Prod mode are paired.

9.7.4.2. Create projects

A project is used to isolate physical resources and developers during Data Mid-End construction. This topic describes how to create projects in Dev-Prod mode or a project in Basic mode.

Prerequisites

Required computing engines are created. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.

Context

Limits:

- Only the super administrator and business unit administrators can create projects.
- A computing engine can be bound to only one project.
- You cannot change the computing engine of a project after they are bound.
- To make sure that tasks can be processed in a project in Prod mode, you must configure the stream processing computing engine or batch processing computing engine for both the project in Prod mode and the corresponding project in Dev mode.

Dataphin allows you to create projects in Dev-Prod or Basic mode. The following table describes the two modes.

Mode	Description
Dev-Prod mode	Generates a project in Dev mode as the development environment and a project in Prod mode as the production environment. The two projects are isolated. This allows you to ensure data security in the production environment and manage the data production process in a controllable way. We recommend that you select this mode if you are concerned more about management. This mode is applicable to the scenario where you have many developers who have clear responsibilities and you have a large budget for computing and storage. Note By default, the name of the project in Dev mode is suffixed with _dev.
Basic mode	Generates a project in Basic mode as an independent and flexible production environment. In this mode, the data production process is stable and easy-to-use and the production data is controllable. We recommend that you select this mode if you are concerned more about data development efficiency rather than management. This mode is applicable to the scenario where the responsibilities of developers are not clear and you have a limited budget for computing and storage.

Create projects in Dev-Prod mode

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Project Management** in the left-side navigation pane.

4. On the **Project Management** page, click **Create Project** in the upper-right corner.
5. In the **Create Project** dialog box, select **Dev-Prod Mode**.
6. Click **Next**.
7. In the **Create Project** dialog box, set the parameters as required.

Section	Parameter	Description
Name Settings	Common Display Name	The common display name of the projects. Note the following naming rules when you set a name: ○ The common display name can contain letters, digits, underscores (_), and hyphens (-). ○ By default, business unit names are prefixed with **LD_**. Therefore, project names cannot have this prefix. This enables Dataphin to distinguish between logical tables and physical tables in data queries. When logical tables and physical tables are referenced, the former is prefixed with the corresponding business unit name, and the latter is prefixed with the corresponding project name. If project names also start with **LD_**, Dataphin cannot distinguish between physical tables and logical tables that are both prefixed with LD_.
	Common Name	The common name of the projects. Note the following naming rules when you set a name: ○ The common name can contain letters, digits, and underscores (_). ○ If the computing engine type is MaxCompute, we recommend that you set **Common Name** to the MaxCompute project name. ○ By default, business unit names are prefixed with **LD_**. Therefore, project names cannot have this prefix. This enables Dataphin to distinguish between logical tables and physical tables in data queries. When logical tables and physical tables are referenced, the former is prefixed with the corresponding business unit name, and the latter is prefixed with the corresponding project name. If project names also start with **LD_**, Dataphin cannot distinguish between physical tables and logical tables that are both prefixed with LD_.
	Business Unit	Optional. The business unit to which the project belongs.

Section	Parameter	Description
Namespace	Project Type	The type of the project. This parameter is used to categorize the tasks and generated data of projects. Default value: Application Data Store. Valid values: ○ **Source Data**: stores raw data of business databases. This layer serves as the source and basis in subsequent data development and is also called the vertical data center. ○ **Common Dimensional Modeling**: extracts themes, standards, and common data from business data. This layer connects the source data layer and application data store layer and is also called the common data center. ○ **Application Data Store**: defines diversified and distinct metrics based on business scenarios.
Project 1: Dev	Batch Processing	The batch processing computing engine. ○ Make sure that the computing engine that you bind to each project is unique. Otherwise, a conflict may occur when data is written to the physical database. ○ Batch processing computing engines are bound to MaxCompute projects. If you have multiple projects, each bound to a batch processing computing engine, make sure that your AccessKey pair has the permissions of the MaxCompute project administrator, including the permission to access data across the MaxCompute projects. If your AccessKey pair does not have the preceding permissions, you must execute the GRANT statement in the authorization code in MaxCompute to grant required permissions to your AccessKey pair. This ensures that you can pass the database authentication when you switch between projects in Dataphin. ○ A batch processing computing engine can be bound to only one project.
	Stream Processing	The stream processing computing engine.
	Name	Optional. The name of the project in Dev mode. Dataphin automatically generates a name based on the specified common name of the project in the format of Common name_dev.
	Display Name	Optional. The display name of the project in Dev mode. Dataphin automatically generates a name based on the specified common display name of the project in the format of Common display name_development.
	Description	The description of the project in Dev mode.

Section	Parameter	Description
Project 2: Prod	Stream Processing	The stream processing computing engine.
	Batch Processing	The batch processing computing engine. For more information, see the description of the batch processing computing engine in Project 1: Dev. Note You must bind different computing engines to the project in Dev mode and the project in Prod mode.
	Stream Processing	The stream processing computing engine.
	Name	Optional. The name of the project in Prod mode. Dataphin automatically generates a name based on the specified common name of the project.
	Display Name	Optional. The display name of the project in Prod mode. Dataphin automatically generates a name based on the specified common display name of the project.
	Description	The description of the project in Prod mode.
Other Settings	Sandbox Whitelist	The IP addresses, domain names, and database endpoints that Shell or Python tasks can access. To configure a sandbox whitelist, perform the following steps: i. Click **Create**. In the fields that appear, enter the IP address, domain name, or database endpoint and the port number. ii. Click the icon to add the IP address, domain name, or database endpoint to the sandbox whitelist. You can find an IP address, a domain name, or a database endpoint and click the icon in the **Actions** column to remove it from the sandbox whitelist.

8. Click **OK**.

Create a project in Basic mode

1. On the **Project Management** page, click **Create Project** in the upper-right corner.
2. In the **Create Project** dialog box, select **Basic Mode**.
3. Click **Next**.
4. In the **Create Project** dialog box, set the parameters as required.

Section	Parameter	Description
Basic Settings	Batch Processing	The batch processing computing engine. ◦ Make sure that the computing engine that you bind to each project is unique. Otherwise, a conflict may occur when data is written to the physical database. ◦ Batch processing computing engines are bound to MaxCompute projects. If you have multiple projects, each bound to a batch processing computing engine, make sure that your AccessKey pair has the permissions of the MaxCompute project administrator, including the permission to access data across the MaxCompute projects. If your AccessKey pair does not have the preceding permissions, you must execute the GRANT statement in the authorization code in MaxCompute to grant required permissions to your AccessKey pair. This ensures that you can pass the database authentication when you switch between projects in Dataphin. ◦ A batch processing computing engine can be bound to only one project.
	Stream Processing	The stream processing computing engine.
	Name	The name of the project. Note the following naming rules when you set a name: ◦ The name can contain letters, digits, and underscores (_). ◦ By default, business unit names are prefixed with LD_. Therefore, project names cannot have this prefix. This enables Dataphin to distinguish between logical tables and physical tables in data queries. When logical tables and physical tables are referenced, the former is prefixed with the corresponding business unit name, and the latter is prefixed with the corresponding project name. If project names also start with LD_, Dataphin cannot distinguish between physical tables and logical tables that are both prefixed with LD_.

Section	Parameter	Description
Namespace	**Display Name**	The display name of the project. Note the following naming rules when you set a name: ○ The display name can contain letters, digits, underscores (_), and hyphens (-). ○ By default, business unit names are prefixed with **LD_**. Therefore, project names cannot have this prefix. This enables Dataphin to distinguish between logical tables and physical tables in data queries. When logical tables and physical tables are referenced, the former is prefixed with the corresponding business unit name, and the latter is prefixed with the corresponding project name. If project names also start with **LD_**, Dataphin cannot distinguish between physical tables and logical tables that are both prefixed with LD_.
	Description	The description of the project in Basic mode.
	Business Unit	Optional. The business unit to which the project belongs.
	Project Type	The type of the project. This parameter is used to categorize the tasks and generated data of projects. Default value: Application Data Store. Valid values: ○ **Source Data**: stores raw data of business databases. This layer serves as the source and basis in subsequent data development and is also called the vertical data center. ○ **Common Dimensional Modeling**: extracts themes, standards, and common data from business data. This layer connects the source data layer and application data store layer and is also called the common data center. ○ **Application Data Store**: defines diversified and distinct metrics based on business scenarios.

Section	Parameter	Description
Other Settings	Sandbox Whitelist	The IP addresses, domain names, and database endpoints that Shell or Python tasks can access. To configure a sandbox whitelist, perform the following steps: i. Click **Create**. In the fields that appear, enter the IP address, domain name, or database endpoint and the port number. ii. Click the icon to add the IP address, domain name, or database endpoint to the sandbox whitelist. You can find an IP address, a domain name, or a database endpoint and click the icon in the **Actions** column to remove it from the sandbox whitelist.

5. Click **OK**.

9.7.4.3. Configure projects

You can configure project information and manage project members. This topic describes how to configure projects in Dev-Prod mode or a project in Basic mode.

Prerequisites

A project in Basic mode is created or projects in Dev-Prod mode are created. For more information, see Create projects.

Context

Limits:

- By default, the super administrator also plays the administrator role in all projects. You cannot delete or change the super administrator role.
- Only project administrators can change the roles of other project members.
- Only the super administrator and project administrators can configure project information.

Configure the information about projects in Dev-Prod mode

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Project Management** in the left-side navigation pane.
4. On the Project Management page, find the project that you want to configure and click **Information Settings**.
5. In the **Information Settings** dialog box, change the parameter values as needed and set the

parameters in the General Feature Control section. For more information, see Create projects.

- If the **Data Result Download** parameter is set to **Yes**, the roles except the guests can download result data.
- If the **Data Result Download** parameter is set to **No**, the result data cannot be downloaded and can only be viewed on the Result tab.

6. Click **OK**.

Configure the information about a project in Basic mode

1. On the **Project Management** page, find the project that you want to configure and click **Information Settings**.
2. In the **Information Settings** dialog box, change the parameter values as needed and set the parameters in the **General Feature Control** section. For more information, see Create projects.
   - If the **Data Result Download** parameter is set to **Yes**, the roles except the guests can download result data.
   - If the **Data Result Download** parameter is set to **No**, the result data cannot be downloaded and can only be viewed on the Result tab.
3. Click **OK**.

Add members

1. On the **Project Management** page, find the project that you want to configure and click **Members Management**.
2. In the **Manage Members** dialog box, click **Add Members**.
3. Add one or more members.
   i. In the **Add Members** dialog box, set the parameters as required.

Parameter	Description
Username	The user to be added as a project member. You can specify multiple users.
Role	The role that you want the selected users to assume.

   ii. Click **OK**.
4. After you set the preceding parameters, click **OK**.

Change the role of a member

1. On the **Project Management** page, find the project that you want to configure and click **Members Management**.
2. In the **Manage Members** dialog box, find the member whose role you want to change and click the icon in the **Role** column.
3. From the drop-down list, select a role.
4. Click **OK**.

Remove a member

1. On the **Project Management** page, find the project that you want to configure and click **Members Management**.
2. Remove a member.
   i. In the **Manage Members** dialog box, find the member you want to remove and click the icon in the **Actions** column.
   ii. In the message that appears, click **OK**.
3. In the Manage Members dialog box, click **OK**.

Remove multiple members at a time

1. On the **Project Management** page, find the project that you want to configure and click **Members Management**.
2. Remove multiple members at a time.
   i. In the **Manage Members** dialog box, select multiple members to be removed.
   ii. Click **Remove** in the lower part of the **Manage Members** dialog box.
   iii. In the message that appears, click **OK**.
3. In the Manage Members dialog box, click **OK**.

9.7.4.4. Delete projects

A project is used to isolate physical resources and developers during Data Mid-End construction. This topic describes how to delete projects.

Prerequisites

A project in Basic mode is created or projects in Dev-Prod mode are created. For more information, see Create projects.

Context

Limits:

- Only the super administrator and project administrators can delete projects.
- Only projects that contain no tasks or objects, such as code tasks or data standardization and data modeling objects, can be deleted.
- To delete projects in Dev-Prod mode, make sure that the project in Dev mode and the project in Prod mode contain no tasks or objects.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the Planning page, click **Project Management** in the left-side navigation pane.

4. In the lower part of the **Project Management** page, click **Enable Batch Deletion**.
5. Select the projects to be deleted. To delete projects in Dev-Prod mode, you can select only the project in Dev mode or the project in Prod mode. The other project is automatically selected.
6. In the lower part of the page, click **Delete**.
7. In the message that appears, click **OK**.

9.7.5. Maintenance and upgrade

9.7.5.1. Important notes for upgrade

This topic describes the important notes for upgrading business units and projects.

If you want to upgrade Dataphin or control Dataphin for data security purposes, you can lock Dataphin by setting Dataphin to work in the maintenance mode.

When the business grows or the publishing process of some data must be controlled, you can upgrade business units or projects from the Basic mode to the Dev-Prod mode. During maintenance and upgrade, data read and write operations are not allowed, but tasks run normally.

Important notes for upgrading business units

- Only the super administrator can upgrade business units.
- You cannot downgrade business units after they are upgraded.
- You cannot upgrade business units whose names are suffixed with _dev.
- You can upgrade only one business unit at a time.
- If you upgrade a business unit in the Basic mode that has no projects, the business unit is upgraded to a business unit in the Prod mode. A corresponding business unit in the Dev mode is created. The two business units share all configurations and resources.
- If you upgrade a business unit in the Basic mode that has projects with data standardization and data modeling objects, the projects are forcibly upgraded to projects in the Prod mode. Corresponding projects in the Dev mode are created.
- If you upgrade a business unit in the Basic mode that has projects without data standardization or data modeling objects, the projects remain in the Basic mode. However, the data standardization and data modeling features of these projects are disabled during and after the upgrade. For example, when you open a logical aggregate table found by global search in these projects, you can only view its information, but cannot perform any operations.

Important notes for upgrading projects

- Only the super administrator can upgrade projects.
- You cannot downgrade projects after they are upgraded.
- You can upgrade one or more projects at a time.
- If you upgrade a project in the Basic mode that is bound to a business unit, the project is upgraded to a project in the Prod mode. A corresponding project in the Dev mode is created. The data standardization and data modeling features are enabled for both projects after the upgrade.
- If you upgrade a project in the Basic mode, Dataphin upgrades the project to a project in the

Prod mode and copies the configuration of the original project in the Basic mode to create a project in the Dev mode. Objects in the draft state in the original project are automatically deleted.

- After a project in the Basic mode is upgraded, you can publish objects in the project in the Dev mode to overwrite the corresponding objects in the developing state in the project in the Prod mode.

9.7.5.2. Maintain and upgrade business units

This topic describes how to maintain and upgrade business units.

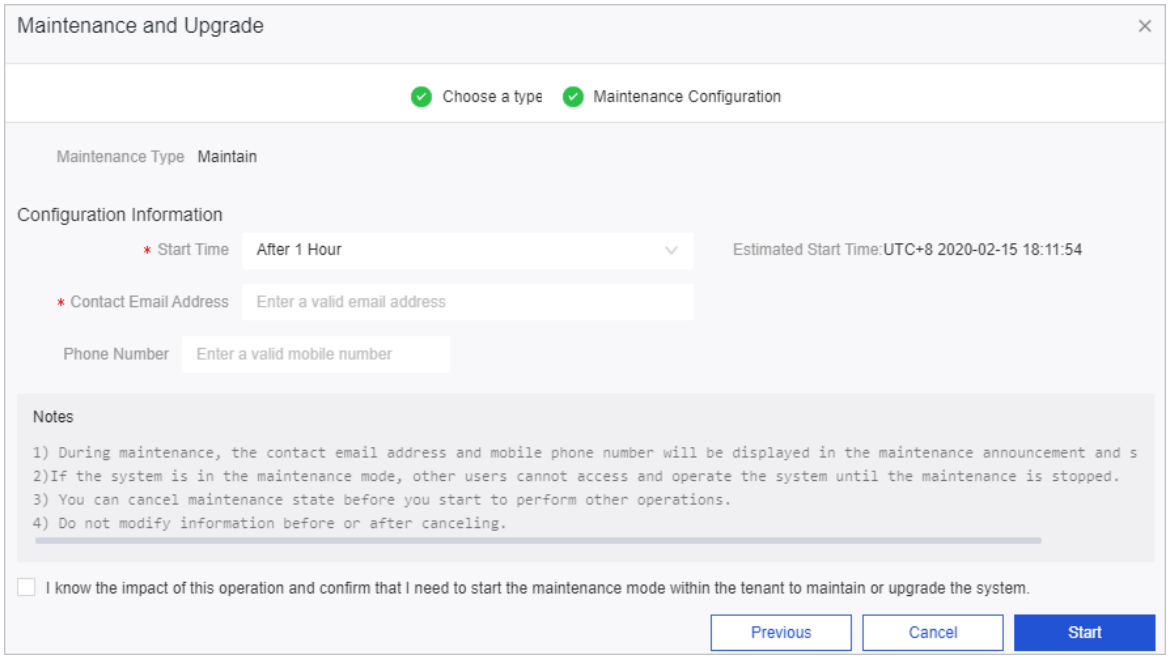
Maintain business units

1. Log on to the Dataphin console.
2. Open the **Maintenance and Upgrade** dialog box by using one of the following methods:
   - a. On the Dataphin homepage, click **Planning** in the top navigation bar.

     You can also click **Intelligent Data Warehouse Planning** in the middle section.

     b. On the **Planning** page, click **Business Units** in the left-side navigation pane.

     c. On the **Business Units** page, click **Maintenance and Upgrade** in the upper-right corner.
   - On the Dataphin homepage, move the pointer over the icon in the upper-right corner and select **Start Maintenance and Upgrade**.
3. In the **Maintenance and Upgrade** dialog box, set the **Choose a type** parameter to **Maintain**. Click **Next**.
4. In the Maintenance Configuration step, set the **Start Time**, **Phone Number**, and **Contact Email Address** parameters, select **I know the impact of this operation and confirm that I need to start the maintenance mode within the tenant to maintain or upgrade the system**, and then click **Start**.



If you want to cancel the maintenance before the maintenance starts, perform the following steps:

i. In the **Maintenance and Upgrade** dialog box, click **Cancel Maintenance**.

ii. In the message that appears, click **OK**.

After the maintenance starts, you cannot perform operations on modules. If you want to stop the ongoing maintenance, perform the following steps as the super administrator:

i. Go to the **Starting Maintenance** page and click **Forcedly Stop Maintenance**.

ii. In the message that appears, click **OK**.

Upgrade business units

1. In the **Maintenance and Upgrade** dialog box, set the **Choose a type** parameter to **Upgrade to Dev-Prod Mode**. Click **Next**.
2. In the Maintenance Configuration step, set the **Start Time**, **Phone Number**, and **Contact Email Address** parameters, select **I know the impact of this operation and confirm that I need to start the maintenance mode within the tenant to maintain or upgrade the system**, and then click **Start**.

   If you want to cancel the upgrade before the upgrade starts, perform the following steps:

   i. In the **Maintenance and Upgrade** dialog box, click **Cancel Maintenance**.

   ii. In the message that appears, click **OK**.

   After the upgrade starts, you cannot perform operations on modules. If you want to stop the ongoing upgrade, perform the following steps as the super administrator:

   i. On the left of the Upgrade Configuration page, click **Stop Maintenance**.

   ii. In the message that appears, click **OK**.
3. After the upgrade is completed, click **Upgraded**.
4. On the page that appears, click **Continue to Upgrade** or **Exit Maintenance Mode**.

9.7.5.3. View maintenance and upgrade logs

This topic describes how to view maintenance and upgrade logs.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, move the pointer over the  icon in the upper-right corner and select **View History**.
3. In the **Operations Log for Maintenance and Upgrades** dialog box, view the operators, operation time, and operations.
4. Click **OK**.

9.7.6. Data source

9.7.6.1. Overview

Data source overview

A data source is a physical data store. You can create data sources to import business data to Dataphin, based on which you can build a Data Mid-End. You can also import existing data to a data source.

Data source types

The following table describes the types of data sources that Dataphin supports.

Category	Data source type
Batch processing	MaxCompute, MySQL, SQL Server, PostgreSQL, Oracle, Hadoop Distributed File System (HDFS), Hive, FTP, Vertica, PolarDB-X, AnalyticDB, Elasticsearch, HBase_1_1_X, HBase_0_9_4, MongoDB, AnalyticDB for MySQL V3.0, AnalyticDB for PostgreSQL, and LogHub
Stream processing	PolarDB-X, DataHub, ApsaraDB for HBase, Log Service, Tablestore, Kafka_9_11, and RocketMQ

Access methods

Dataphin can use different methods to connect to different types of databases.

- Non-ApsaraDB databases accessible from the Internet: After Dataphin is granted the access to a non-ApsaraDB database accessible from the Internet, you can connect Dataphin to the database by using the public IP address or endpoint of the database.
- ApsaraDB databases: If an ApsaraDB database is accessible from the Internet, you can connect Dataphin to the database by using the public IP address of the database after Dataphin is granted the access to the database. If an ApsaraDB database is in a virtual private cloud (VPC), you can connect Dataphin to the database by using the VPC after the IP address of Dataphin is added to the whitelist of the database.
- User-created databases on ECS instances in VPCs: After the IP address of Dataphin is added to the whitelist of a user-created database on an ECS instance in a VPC, you can connect Dataphin to the database by using the VPC.
- Databases in a MaxCompute project: If a database is used in a MaxCompute project, you can connect Dataphin to the database by using the endpoint of the database.

Permissions

- All Dataphin members can create data sources.
- The super administrator can delete and modify all data sources, test the connectivity of all data sources, and change the owners of all data sources.
- Members other than the super administrator can only delete and modify their own data sources, test the connectivity of their own data sources, and change the owner of their own data sources.

9.7.6.2. Create batch processing data sources

9.7.6.2.1. Create a MaxCompute data source

This topic describes how to create a MaxCompute data source.

Prerequisites

- The endpoint that you can use to connect to MaxCompute is obtained.
- The name of a MaxCompute project is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the MaxCompute project are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
   - The **Data Sources** page displays the information about all created data sources, including the name, owner, supported data processing type, creation information, and connection information of each data source.
   - In the upper-right corner of the **Data Sources** page, you can enter a keyword in the search box to search for a specific data source by name.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **MAX_COMPUTE**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ◦ If you want to use the data source only in the production environment, select **Prod**. ◦ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ▪ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ▪ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to the MaxCompute project.
Project Name	The name of the MaxCompute project to which the data source is connected.

Parameter	Description
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the MaxCompute project.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the MaxCompute project.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.2. Create a MySQL data source

This topic describes how to create a MySQL data source.

Prerequisites

- The JDBC URL that you can use to connect to a MySQL database is obtained.
- The username and password that you can use to connect to the MySQL database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the Planning page, click **Data Sources** in the left-side navigation pane.
   - The **Data Sources** page displays the information about all created data sources, including the name, owner, supported data processing type, creation information, and connection information of each data source.
   - In the upper-right corner of the **Data Sources** page, you can enter a keyword in the search box to search for a specific data source by name.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **MYSQL**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the MySQL database.
Username	The username that you can use to connect to the MySQL database.
Password	The password that you can use to connect to the MySQL database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.3. Create an SQL Server data source

This topic describes how to create an SQL Server data source.

Prerequisites

- The JDBC URL that you can use to connect to an SQL Server database is obtained.
- The schema of the SQL Server database is obtained.
- The username and password that you can use to connect to the SQL Server database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **SQL_SERVER**.

Parameter	Description
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the SQL Server database.
Schema	The schema of the SQL Server database.
Username	The username that you can use to connect to the SQL Server database.
Password	The password that you can use to connect to the SQL Server database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.4. Create a PostgreSQL data source

This topic describes how to create a PostgreSQL data source.

Prerequisites

- The JDBC URL that you can use to connect to a PostgreSQL database is obtained.
- The schema of the PostgreSQL database is obtained.
- The username and password that you can use to connect to the PostgreSQL database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.

4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **POSTGRE_SQL**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the PostgreSQL database.
Schema	The schema of the PostgreSQL database.
Username	The username that you can use to connect to the PostgreSQL database.
Password	The password that you can use to connect to the PostgreSQL database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.5. Create an Oracle data source

This topic describes how to create an Oracle data source.

Prerequisites

- The JDBC URL that you can use to connect to an Oracle database is obtained.
- The schema of the Oracle database is obtained.
- The username and password that you can use to connect to the Oracle database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar

- On the Dataphin homepage, click **Planning** in the top navigation bar.
- On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.

3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ORACLE**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the Oracle database.
Schema	The schema of the Oracle database.
Username	The username that you can use to connect to the Oracle database.
Password	The password that you can use to connect to the Oracle database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.6. Create an HDFS data source

This topic describes how to create an HDFS data source.

Prerequisites

- The JDBC URL that you can use to connect to HDFS is obtained.
- The URL that you can use to connect to an HDFS file system is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **HDFS**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
DefaultFS	The URL of the HDFS file system, in the format of hdfs://Server IP:Port.
Kerberos	Specifies whether to enable Kerberos authentication. You can select **Enable** or **Disable**. If you select **Enable**, you must also set the following parameters: ○ **KDC Server**: the endpoint of the key distribution center (KDC) server. Separate multiple endpoints with commas (,). ○ **Keytab File**: the keytab file for Kerberos authentication. ○ **Principal**: the Kerberos principal name for Kerberos authentication.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.7. Create an FTP data source

This topic describes how to create an FTP data source.

Prerequisites

- The IP address of an FTP server is obtained.
- The port number of the FTP server is obtained.
- The username and password that you can use to connect to the FTP server are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **FTP**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Protocol	The protocol that is used to transfer files. You can select **FTP** or **SFTP**. ○ **FTP**: transfers files by using FTP. ○ **SFTP**: transfers files by using SFTP.
Host	The name of the FTP server.
Port	The port number of the FTP server.
Username	The username that you can use to connect to the FTP server.

Parameter	Description
Password	The password that you can use to connect to the FTP server.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.8. Create a Hive data source

This topic describes how to create a Hive data source.

Prerequisites

- The following cluster configurations of Hive are obtained:
  - The hostname of the Hive cluster
  - The IP address of the Hive cluster
  - The port number of the Hive cluster
- The following HDFS configurations of Hive are obtained:
  - The keytab file for Kerberos authentication for HDFS
  - The Kerberos principal name for Kerberos authentication for HDFS
- The following Hive configurations are obtained:
  - The JDBC URL that you can use to connect to Hive
  - The keytab file and the Kerberos principal name for Kerberos authentication for Hive
- The following configurations of the metadatabase are obtained:
  - **JDBC URL**
  - The username that you can use to connect to the metadatabase
  - The password that you can use to connect to the metadatabase

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Section	Parameter	Description
	Type	The type of the data source. Select **HIVE**.

Section	Parameter	Description
Basic information	Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
	Description	The description of the data source.
	Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Cluster Configuration	NameNode	The location of the current file system root, in the format of Hostname:Port or IP address:Port. Specify the hostname, port number, and IP address of the Hive cluster.
	Kerberos	Specifies whether to enable Kerberos authentication. You can select **Enable** or **Disable**. If you select **Enable**, you must set the **KDC Server** parameter or upload the Kerberos configuration file. The **KDC Server** parameter specifies the endpoint of the KDC server. Separate multiple endpoints with commas (,).
	KRB5 File	The Kerberos configuration file that is used for authentication.
HDFS Configuration	HDFS Keytab File	The keytab file for Kerberos authentication for HDFS.
	HDFS Principal	The Kerberos principal name for Kerberos authentication for HDFS.
Hive Configuration	JDBC URL	The JDBC URL that you can use to connect to Hive.
	Hive Keytab File	The keytab file for Kerberos authentication for Hive.
	Hive Principal	The Kerberos principal name for Kerberos authentication for Hive.

Section	Parameter	Description
Metadatabase Configuration	Database Type	The type of the metadatabase. Valid values: **MySQL** and **PostgreSQL**.
	JDBC URL	The JDBC URL that you can use to connect to the metadatabase.
	Username	The username that you can use to connect to the metadatabase.
	Password	The password that you can use to connect to the metadatabase.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.9. Create an Elasticsearch data source

This topic describes how to create an Elasticsearch data source.

Prerequisites

- The URL that you can use to connect to Elasticsearch is obtained.
- The username and password that you can use to connect to Elasticsearch are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ELASTIC_SEARCH**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
ES URL	The URL that you can use to connect to Elasticsearch.
Username	The username that you can use to connect to Elasticsearch.
Password	The password that you can use to connect to Elasticsearch.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.10. Create a MongoDB data source

This topic describes how to create a MongoDB data source.

Prerequisites

- The JDBC URL that you can use to connect to a MongoDB database is obtained.
- The username and password that you can use to connect to the MongoDB database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **MONGODB**.

Parameter	Description
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the MongoDB database.
Username	The username that you can use to connect to the MongoDB database.
Password	The password that you can use to connect to the MongoDB database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.11. Create an AnalyticDB data source

This topic describes how to create an AnalyticDB data source.

Prerequisites

- The JDBC URL that you can use to connect to an AnalyticDB database is obtained.
- The username and password that you can use to connect to the AnalyticDB database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ANALYTICDB**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the AnalyticDB database.
Username	The username that you can use to connect to the AnalyticDB database.
Password	The password that you can use to connect to the AnalyticDB database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.12. Create a PolarDB-X data source

This topic describes how to create a PolarDB-X data source.

Prerequisites

- The JDBC URL that you can use to connect to a PolarDB-X database is obtained.
- The username and password that you can use to connect to the PolarDB-X database are obtained.

Procedure

1. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
2. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
3. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
4. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **DRDS**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the PolarDB-X database.
Username	The username that you can use to connect to the PolarDB-X database.
Password	The password that you can use to connect to the PolarDB-X database.

5. Click **Test Connection** to test the connectivity of the data source.
6. After the data source passes the connectivity test, click **OK**.

9.7.6.2.13. Create a Vertica data source

This topic describes how to create a Vertica data source.

Prerequisites

- The JDBC URL that you can use to connect to a Vertica database is obtained.
- The schema of the Vertica database is obtained.
- The username and password that you can use to connect to the Vertica database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.

- The **Data Sources** page displays the information about all created data sources, including the name, owner, supported data processing type, creation information, and connection information of each data source.
- In the upper-right corner of the **Data Sources** page, you can enter a keyword in the search box to search for a specific data source by name.

4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **VERTICA**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the Vertica database.
Schema	The schema of the Vertica database.
Username	The username that you can use to connect to the Vertica database.
Password	The password that you can use to connect to the Vertica database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.14. Create an HBase data source

This topic describes how to create an HBase data source.

Prerequisites

- The endpoints of KDC servers are obtained.
- The Kerberos principal name for Kerberos authentication is obtained.
- The keytab file for Kerberos authentication is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **HBASE_0_9_4** or **HBASE_1_1_X**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to an HBase database.
Connection Parameters	The parameters that are required to connect to the HBase database. Enter the parameters in the JSON format.
Kerberos	Specifies whether to enable Kerberos authentication. You can select **Enable** or **Disable**. If you select **Enable**, you must also set the following parameters: ○ **KDC Server**: the endpoint of the KDC server. Separate multiple endpoints with commas (,). ○ **Keytab File**: the keytab file for Kerberos authentication. ○ **Principal**: the Kerberos principal name for Kerberos authentication.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.15. Create an AnalyticDB for MySQL V3.0 data source

This topic describes how to create an AnalyticDB for MySQL V3.0 data source.

Prerequisites

- The JDBC URL that you can use to connect to an AnalyticDB for MySQL V3.0 database is obtained.
- The username and password that you can use to connect to the AnalyticDB for MySQL V3.0 database are obtained.
- The ID of the cluster where the AnalyticDB for MySQL V3.0 database resides is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ADB_FOR_MYSQL_V3**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ◦ If you want to use the data source only in the production environment, select **Prod**. ◦ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ▪ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ▪ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the AnalyticDB for MySQL V3.0 database.

Parameter	Description
Username	The username that you can use to connect to the AnalyticDB for MySQL V3.0 database.
Password	The password that you can use to connect to the AnalyticDB for MySQL V3.0 database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.16. Create an AnalyticDB for PostgreSQL data source

This topic describes how to create an AnalyticDB for PostgreSQL data source.

Prerequisites

- The JDBC URL that you can use to connect to an AnalyticDB for PostgreSQL database is obtained.
- The schema of the AnalyticDB for PostgreSQL database is obtained.
- The username and password that you can use to connect to the AnalyticDB for PostgreSQL database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ADB_FOR_PG**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
JDBC URL	The JDBC URL that you can use to connect to the AnalyticDB for PostgreSQL database.
Schema	The schema of the AnalyticDB for PostgreSQL database.
Username	The username that you can use to connect to the AnalyticDB for PostgreSQL database.
Password	The password that you can use to connect to the AnalyticDB for PostgreSQL database.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.2.17. Create a LogHub data source

This topic describes how to create a LogHub data source.

Prerequisites

- The endpoint that you can use to connect to LogHub is obtained.
- The name of a LogHub project is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the LogHub project are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.

5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **LOG_HUB**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
LogHub Endpoint	The endpoint that you can use to connect to LogHub.
Project	The name of the LogHub project.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the LogHub project.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the LogHub project.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3. Create stream processing data sources

9.7.6.3.1. Create a DataHub data source

This topic describes how to create a DataHub data source.

Prerequisites

- The endpoint that you can use to connect to DataHub is obtained.
- The name of a DataHub project is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the DataHub project are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **DATAHUB**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to DataHub.
Project Name	The name of the DataHub project.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the DataHub project.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the DataHub project.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3.2. Create a Log Service data source

This topic describes how to create a Log Service data source.

Prerequisites

- The endpoint that you can use to connect to Log Service is obtained.
- The name of a Log Service project is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the Log Service project are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **LOG_SERVICE**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to Log Service.
Project Name	The name of the Log Service project.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the Log Service project.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the Log Service project.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3.3. Create an ApsaraDB for HBase data source

This topic describes how to create an ApsaraDB for HBase data source.

Prerequisites

- The endpoint that you can use to connect to the quorum servers of ZooKeeper built in ApsaraDB for HBase is obtained.
- The endpoint that you can use to connect to an ApsaraDB for HBase database is obtained.
- The username and password that you can use to connect to the ApsaraDB for HBase database are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ALIYUN_HBASE**.
Name	The name of the data source.
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.

Parameter	Description
Service type	The edition of ApsaraDB for HBase. Valid values: **STANDARD** and **ENHANCED**. ○ If you set the **Service type** parameter to **STANDARD**, you must also set the **Version** and **zkQuorum** parameters. ○ If you set the **Service type** parameter to **ENHANCED**, you must also set the **endPoint**, **Username**, and **Password** parameters.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3.4. Create a Kafka data source

This topic describes how to create a Kafka data source.

Prerequisites

The endpoint that you can use to connect to a Kafka cluster is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **KAFKA_9_11**.
Name	The name of the data source.
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Kafka cluster address	The endpoint that you can use to connect to the Kafka cluster.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3.5. Create a Tablestore data source

This topic describes how to create a Tablestore data source.

Prerequisites

- The endpoint that you can use to connect to a Tablestore instance is obtained.
- The name of the Tablestore instance is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the Tablestore instance are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **TABLE_STORE**.
Name	The name of the data source.
Description	The description of the data source.

Parameter	Description
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to the Tablestore instance.
Access Id	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the Tablestore instance.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the Tablestore instance.
Instance Name	The name of the Tablestore instance.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.3.6. Create a RocketMQ data source

This topic describes how to create a RocketMQ data source.

Prerequisites

- The endpoint that you can use to connect to a RocketMQ instance is obtained.
- The ID of the RocketMQ instance is obtained.
- The AccessKey ID and AccessKey secret that are used to authenticate the user identity when Dataphin connects to the RocketMQ instance are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the **Data Sources** page, click **Create Data Source** in the upper-right corner.
5. In the **Create Data Source** dialog box, set the parameters as required.

Parameter	Description
Type	The type of the data source. Select **ROCKET_MQ**.
Name	The name of the data source. The name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the data source.
Environment	The type of the environment where the data source will be used. ○ If you want to use the data source only in the production environment, select **Prod**. ○ If you want to use the data source in both the production and development environments, create the data source by using one of the following methods: ■ Select **Prod** and configure the data source for the production environment. After that, go to the Data Sources page, find the data source, and then click **Data Source for Dev Environment**. ■ Select **Prod and Dev** and configure the data source for both the production and development environments.
Endpoint	The endpoint that you can use to connect to the RocketMQ instance.
InstanceID	The ID of the RocketMQ instance.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the RocketMQ instance.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the RocketMQ instance.

6. Click **Test Connection** to test the connectivity of the data source.
7. After the data source passes the connectivity test, click **OK**.

9.7.6.4. Modify a data source

This topic describes how to modify a data source.

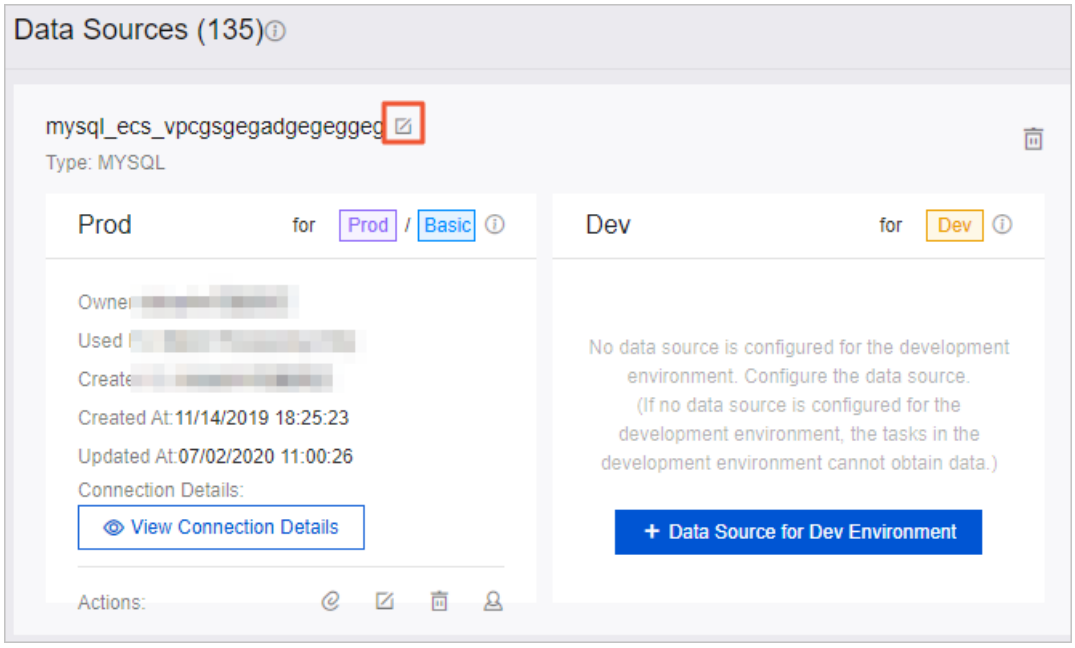
Prerequisites

Data sources are created. For more information, see Create batch processing data sources or Create stream processing data sources.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.

3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. Change the name of a data source.
   i. On the Data Sources page, find the data source that you want to modify and click the icon next to the name of the data source.



   ii. In the field that appears, enter a new name.
5. Modify the connection information about a data source.
   i. Find the data source that you want to modify and click the icon next to **Actions** in the Prod or Dev section.
   ii. In the **Edit Data Source for Prod Environment** or **Edit Data Source for Dev Environment** dialog box, change parameter values as needed. For more information about how to set the parameters, see Create batch processing data sources or Create stream processing data sources.

> **Note** If the data source is bound to a Dataphin project, you cannot change the project name, data source type, or endpoint.

   iii. Click **Test Connection** to test the connectivity of the data source.
   iv. After the data source passes the connectivity test, click **OK**.

9.7.6.5. Test a data source

This topic describes how to test a data source.

Prerequisites

Data sources are created. For more information, see Create batch processing data sources or Create stream processing data sources.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the Data Sources page, find the data source that you want to test and click the icon next to **Actions** in the Prod or Dev section.

9.7.6.6. Change the owner of a data source

This topic describes how to change the owner of a data source.

Prerequisites

Data sources are created. For more information, see Create batch processing data sources or Create stream processing data sources.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. On the Data Sources page, find the data source that you want to change the owner and click the icon next to **Actions** in the Prod or Dev section.
5. In the **Change Owner** dialog box, select the member to whom you want to transfer the ownership of the data source.
6. Click **OK**.

9.7.6.7. Delete a data source

This topic describes how to delete a data source.

Prerequisites

Data sources are created. For more information, see Create batch processing data sources or Create stream processing data sources.

Context

You cannot delete a data source that is bound to a Dataphin project.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Data Sources** in the left-side navigation pane.
4. Delete a data source.
   - On the Data Sources page, find the data source that you want to delete and click the icon numbered 2 or 3 in the preceding figure, next to **Actions** in the Prod or Dev section.
   - On the Data Sources page, click the icon numbered 1 in the preceding figure, next to the name of the data source.
5. In the Delete Data Source message, click **OK**.
   - If you click the icon numbered 2 or 3 in the preceding figure, the data source is deleted from the production or development environment.
   - If you click the icon numbed 1 in the preceding figure, the data source is deleted from both the production and development environments.

9.7.7. Computing engines

9.7.7.1. Overview

A batch processing computing engine provides storage and computing resources for offline computing and a stream processing computing engine provides computing resources for real-time computing.

Usage notes

A stream processing computing engine uses computing resources to process data in queues in real time. A batch processing computing engine can store and then process large amounts of data.

- Before you create a batch processing computing engine, make sure that the computing engine type for batch processing is set on the Computing Engine Configuration page in Management Center.

  > **Note** For more information, see Set the computing engine type.

- By default, Dataphin sets Flink as the computing engine type for stream processing.
- Dataphin supports the MaxCompute computing engine for batch processing and the Flink computing engine for stream processing.
- Before you develop tasks in a Dataphin project, you must bind at least one computing engine to the project. By default, a batch processing computing engine is used to store and then process large amounts of data. You can also use a stream processing computing engine to process data in real time.

- Do not create computing engines based on the same connection information. In addition, we recommend that you do not bind a computing engine to multiple Dataphin projects. Otherwise, a conflict may occur when data is written to the physical database.

Permissions

- Only the super administrator, project administrators, and business unit administrators can create computing engines.
- The super administrator can delete and modify all computing engines, test the connectivity of all computing engines, and change the owners of all computing engines.
- Members other than the super administrator can only delete and modify their own computing engines, test the connectivity of their own computing engines, and change the owner of their own computing engines.

9.7.7.2. Create a batch processing computing engine

This topic describes how to create a batch processing computing engine.

Prerequisites

- The computing engine type for batch processing is set. For more information, see Set the computing engine type.
- The endpoint that you can use to connect to MaxCompute is obtained.
- The name of the MaxCompute project to which the computing engine is connected is obtained.
- The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the MaxCompute project is obtained.
- The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the MaxCompute project is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, move the pointer over **Create Computing Engine** and select **Batch Processing Computing Engine** in the upper-right corner.
5. In the **Batch Processing Computing Engine** dialog box, set the parameters as required.

Parameter	Description
Computing Engine Type	The type of the computing engine. Default value: MaxCompute. You cannot modify this parameter.
Computing Engine Name	The name of the computing engine.

Parameter	Description
Description	The description of the computing engine.
Endpoint	The endpoint that you can use to connect to MaxCompute. By default, the endpoint is set to the endpoint that is specified on the Computing Engine Configuration page in Management Center. You cannot modify this parameter.
Project Name	The name of the MaxCompute project to which the computing engine is connected.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the MaxCompute project. Contact the super administrator to obtain the AccessKey ID.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the MaxCompute project. Contact the super administrator to obtain the AccessKey secret.

> **Note**
>
> - You must specify a MaxCompute project that belongs to the same Apsara Stack tenant account as the current account.
> - To ensure a successful connection to the MaxCompute project, we recommend that you specify the AccessKey ID and AccessKey secret of the MaxCompute project administrator.
> - To ensure normal metadata acquisition, we recommend that you do not change the AccessKey ID and AccessKey secret that are used to connect to the MaxCompute project.
> - After a computing engine is created, you cannot change the computing engine type in Dataphin.

6. Click **Test Connection** to test the connectivity of the computing engine.
7. After the computing engine passes the connectivity test, click **Submit**.

9.7.7.3. Create a stream processing computing engine

This topic describes how to create a stream processing computing engine.

Prerequisites

- The name of the Realtime Compute project to which the computing engine is connected is obtained.
- The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the Realtime Compute project is obtained.
- The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the Realtime Compute project is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, move the pointer over **Create Computing Engine** and select **Stream Processing Computing Engine** in the upper-right corner.
5. In the **Stream Processing Computing Engine** dialog box, set the parameters as required.

Parameter	Description
Computing Engine Type	The type of the computing engine. Default value: **FLINK**. You cannot modify this parameter.
Computing Engine Name	The name of the computing engine.
Description	The description of the computing engine.
Project Name	The name of the Realtime Compute project to which the computing engine is connected.
Access ID	The AccessKey ID that is used to authenticate the user identity when Dataphin connects to the Realtime Compute project.
Access Key	The AccessKey secret that is used to authenticate the user identity when Dataphin connects to the Realtime Compute project.

> **Note**
>
> - You must specify a Realtime Compute project that belongs to the same Apsara Stack tenant account as the current account.
> - To ensure a successful connection to the Realtime Compute project, we recommend that you specify the AccessKey ID and AccessKey secret of the Realtime Compute project administrator.
> - To ensure normal metadata acquisition, we recommend that you do not change the AccessKey ID and AccessKey secret that are used to connect to the Realtime Compute project.
> - After a computing engine is created, you cannot change the computing engine type in Dataphin.

6. Click **Test Connection** to test the connectivity of the computing engine.
7. After the computing engine passes the connectivity test, click **Submit**.

9.7.7.4. Test a computing engine

This topic describes how to test a computing engine.

Prerequisites

Computing engines are created. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, find the computing engine that you want to test and click the icon in the **Actions** column.

9.7.7.5. Modify a computing engine

This topic describes how to modify a computing engine.

Prerequisites

Computing engines are created. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, find the computing engine that you want to modify and click the icon in the **Actions** column.
5. In the **Change Data Source (Computing Engine)** dialog box, change the parameter values as needed. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.
6. Click **Test Connection** to test the connectivity of the computing engine.
7. After the computing engine passes the connectivity test, click **Submit**.

9.7.7.6. Change the owner of a computing engine

This topic describes how to change the owner of a computing engine.

Prerequisites

Computing engines are created. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, find the computing engine that you want to change the owner and click the icon in the **Actions** column.
5. In the **Change Owner** dialog box, select the member to whom you want to transfer the ownership of the computing engine.
6. Click **OK**.

9.7.7.7. Delete a computing engine

This topic describes how to delete a computing engine.

Prerequisites

Computing engines are created. For more information, see Create a batch processing computing engine or Create a stream processing computing engine.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Planning** page by using one of the following methods:
   - On the Dataphin homepage, click **Planning** in the top navigation bar.
   - On the Dataphin homepage, click **Intelligent Data Warehouse Planning** in the middle section.
3. On the **Planning** page, click **Computing Engines** in the left-side navigation pane.
4. On the Computing Engines page, find the computing engine that you want to delete and click the icon in the **Actions** column.
5. In the message that appears, click **OK**.

9.8. Data ingestion

Data ingestion is achieved through data synchronization. Data synchronization is the process of importing data from a table in the source database to a table in the target database. For example, importing data of table A in a MySQL database to table B in a PostgreSQL database.

1. Go to the R&D workbench. Log on to the Dataphin console. Click **R&D** in the top navigation bar or **Data Ingestion** on the homepage.
2. Go to the data ingestion page. On the R&D page, click **Data Processing**, and then click the

Sync Tasks submenu. You can move the pointer over an icon to expand its corresponding submenu. On the Sync Tasks page, you can create folders, and create, save, and publish sync tasks.

9.8.1. Overview

Dataphin allows you to use the data integration or data synchronization service to import business data from data sources to Dataphin.

If you purchased a Dataphin instance after April 2020, you can use the data integration service to synchronize data, which is simple, efficient, secure, and reliable.

The data integration service allows you to create multiple sync tasks at a time for database migration and can automatically create a destination table for data synchronization to MaxCompute. This improves the data synchronization efficiency.

9.8.2. Data integration

9.8.2.1. Overview

Data integration is a simple, efficient data synchronization service in Dataphin. It provides powerful data preprocessing capabilities and can synchronize data among heterogeneous data sources at a high speed with high stability.

Background information

When big data analytics applies in more and more industries, numerous data integration requirements emerge. For example, people want a platform that can create multiple sync tasks at a time with simple operations, support heterogeneous data sources, preprocess data from data sources, and optimize sync tasks based on fault tolerance, throttling, and concurrency configurations.

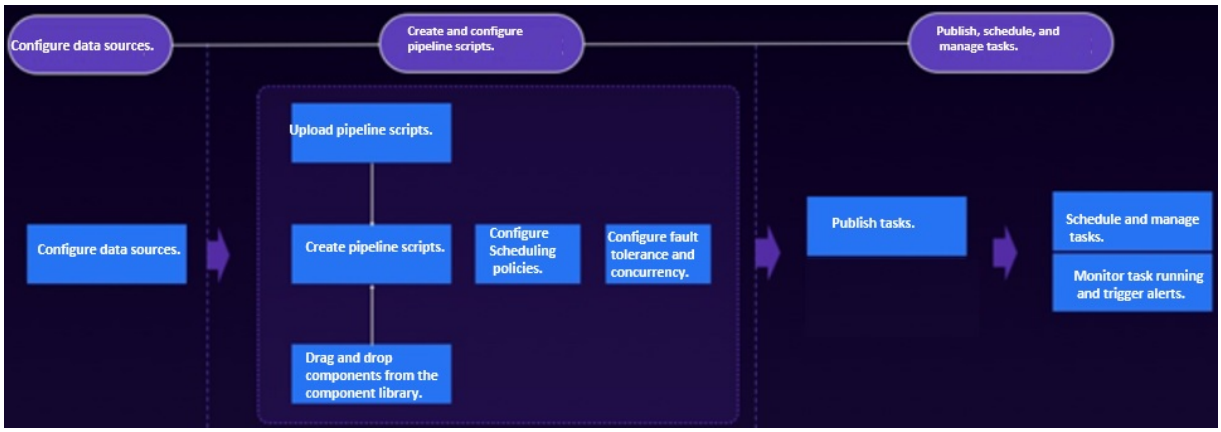
To meet the data integration requirements, Dataphin provides the data integration service with enhanced data integration capabilities. The data integration service is simple, efficient, secure, and reliable and provides the following features:

- Allows you to create multiple sync tasks at a time for database migration and automatically creates a destination table for data synchronization to MaxCompute. This improves the data integration efficiency. For more information, see Create and configure an offline database migration task and Manage Maxcompute output components.
- Provides process and conversion components to preprocess data from data sources, for example, to cleanse, convert, de-identify, compute, merge, distribute, and filter data. For more information, see Create and configure an offline migration pipeline.
- Synchronizes data among heterogeneous data sources at a high speed with high stability. For more information, see Create and configure an offline migration pipeline and Create and configure an offline database migration task.
- Supports projects in Dev-Prod mode and Basic mode.
- Synchronizes logical tables that are created in Dataphin to the destination database.

Features

The data integration service allows you to create pipelines by dragging, configuring, and assembling components. You can also configure pipelines, for example, configure the scheduling policy and pipeline information.

The following figure shows the data integration process in Dataphin.



1. Configure data sources. For more information, see Data sources.
2. Create and configure pipelines for single or multiple sync tasks. For more information, see Create and configure an offline migration pipeline and Create and configure an offline database migration task.
3. Publish tasks. For more information, see Publishing management.
4. Schedule and manage tasks. For more information, see Overview.

9.8.2.2. Manage folders

This topic describes how to create, move, delete, or rename a folder that is used to store pipelines.

Create a folder

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the Script section, click the icon next to **Script**.
6. In the **Create Folder** dialog box, set the **Name** and **Select Directory** parameters.
7. Click **OK**.You can also perform the following steps to create a subfolder under an existing folder:
   i. Move the pointer over a folder and click the icon.
   ii. In the field that appears, enter a folder name.
   iii. Press the Enter key.

Rename a folder

1. In the Script section of the **Integrated** page, move the pointer over the icon next to the folder that you want to rename and select **Rename**.
2. In the field that appears, enter a new folder name.
3. Press the Enter key.

Move a folder

1. In the Script section of the **Integrated** page, move the pointer over the icon next to the folder that you want to move and select **Move**.
2. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory.
3. Click **OK**.

Delete a folder

In the Script section of the **Integrated** page, move the pointer over the icon next to the folder that you want to delete and select **Delete**.

> **Note** You can delete only the folders that do not contain subfolders or items.

9.8.2.3. Upload a JSON file for creating an offline migration pipeline

You can upload a JSON file to Dataphin to create an offline migration pipeline. This topic describes how to upload a JSON file that is used to create an offline migration pipeline.

Prerequisites

A JSON file is prepared.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the **Script** section, click the icon next to **Script**.
6. In the **Upload Script for Creating Pipeline** dialog box, set the parameters as required.

Parameter	Description
Upload File	The JSON file to upload.
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ○ **Recurring Node**: a task that is run on a specified schedule. ○ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

7. Click **OK**.
8. (Optional)Configure the scheduling policy.
   - If you set the **Schedule Type** parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Step 3: Configure the scheduling policy.
   - If you set the **Schedule Type** parameter to **One-Time Node**, you do not need to configure the scheduling policy.
9. Save, submit, and then publish the offline migration pipeline.
   i. Click the icon in the upper-right corner to save the pipeline.
   ii. Click the icon in the upper-right corner to submit the pipeline.
   iii. (Optional)Publish the pipeline.
      - If the current project is in Dev mode, publish the pipeline to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the pipeline after you submit it.

9.8.2.4. Manage offline migration pipelines

9.8.2.4.1. Create and configure an offline migration pipeline

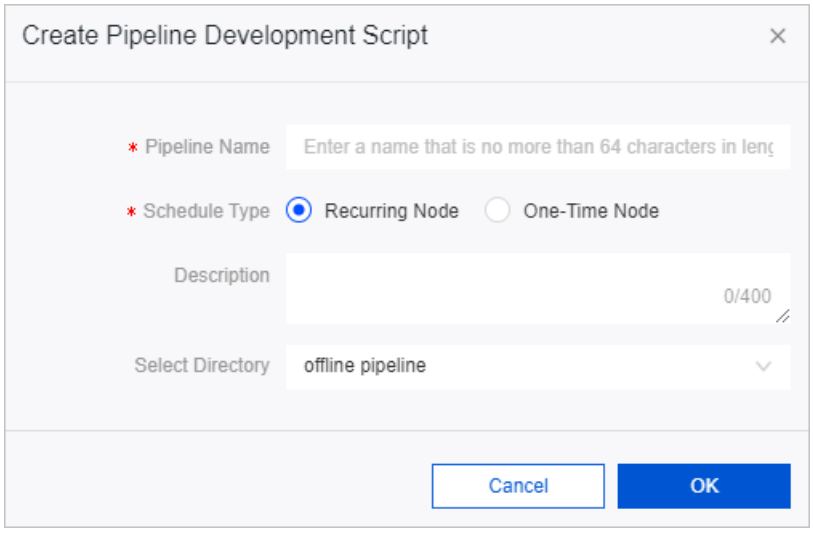
An offline migration pipeline is used to migrate data. This topic describes how to create and configure an offline migration pipeline.

Step 1: Create an offline migration pipeline

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left

corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

4. Move the pointer over **Develop** in the top navigation bar and select **Integrated**.
5. On the **Integrated** page, open the **Create Pipeline Development Script** dialog box by using one of the following methods:
   - In the Script section, move the pointer over the icon next to Script and select **Offline Single Pipeline**.
   - In the left-side navigation pane, move the pointer over the icon next to the project name and select **Offline Single Pipeline**.
6. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



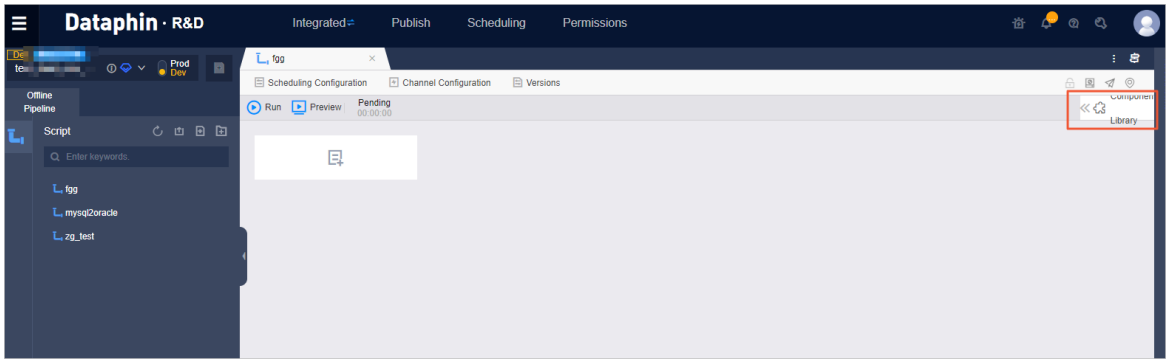
Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ○ **Recurring Node**: a task that is run on a specified schedule. ○ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

7. Click **OK**.

Step 2: Configure components for the offline migration pipeline

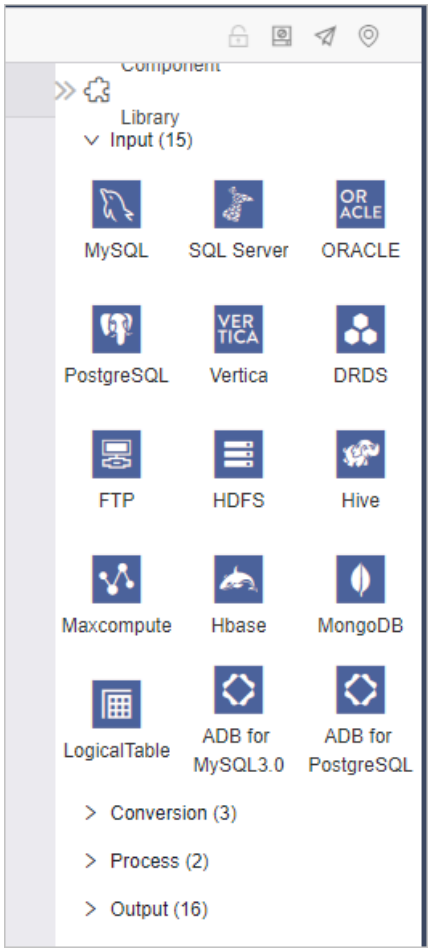
A complete offline migration pipeline is composed of one or more input components, zero or more conversion components, zero or more process components, and one or more output components.

1. Go to the configuration tab of the offline migration pipeline. Click **Component Library** in the upper-right corner. A pane appears and displays the components that Dataphin supports in the **Input**, **Conversion**, **Process**, and **Output** sections.



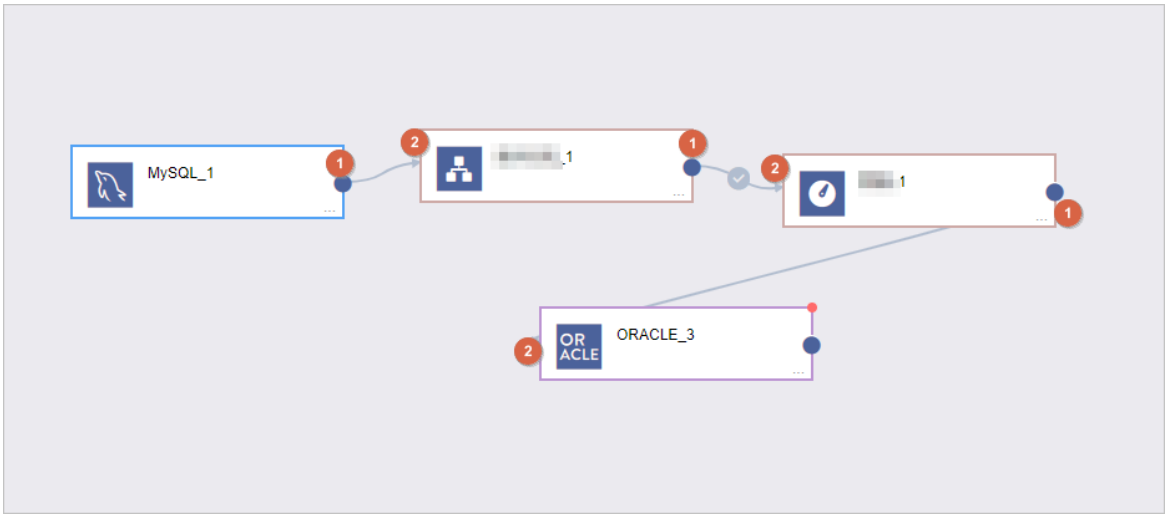
Component type	Component
Input	MySQL, SQL Server, ORACLE, PostgreSQL, Vertica, DRDS, FTP, HDFS, Hive, Maxcompute, Hbase, MongoDB, LogicalTable, ADB for MySQL3.0, and ADB for PostgreSQL
Conversion	Field Selection, Field Computing, and Filtering
Process	Access Limit and Conditional Distribution
Output	MySQL, SQL Server, ORACLE, PostgreSQL, Vertica, DRDS, FTP, HDFS, Hive, Maxcompute, Hbase, MongoDB, ElacticSearch, ADB for MySQL2.0, ADB for MySQL3.0, and ADB for PostgreSQL

2. Select and configure input components based on your business scenario.
   i. Click the > icon before **Input**. Drag a component to the pipeline canvas on the left.
   ii. Right-click the component and select **Configure Attributes**. In the dialog box that appears, configure the component as required. For more information, see Input components. You can also copy or delete the component and select a mode for transmitting data to downstream nodes.

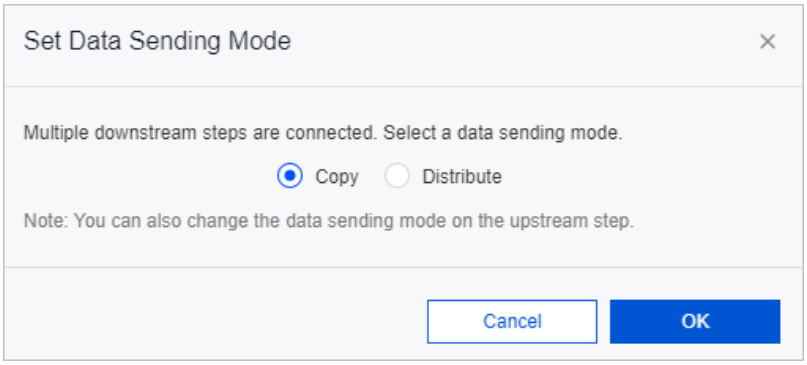


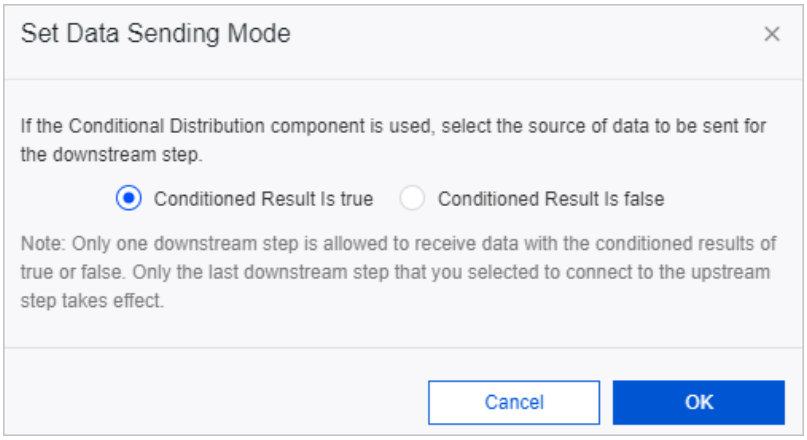
3. (Optional)Select and configure conversion components based on your business scenario.
   i. Click the > icon before **Conversion**. Drag a component to the pipeline canvas on the left.
   ii. Right-click the component and select **Configure Attributes**. In the dialog box that appears, configure the component as required. For more information, see Conversion components. You can also copy or delete the component and select a mode for transmitting data to downstream nodes.
4. (Optional)Select and configure process components based on your business scenario.
   i. Click the > icon before **Process**. Drag a component to the pipeline canvas on the left.
   ii. Right-click the component and select **Configure Attributes**. In the dialog box that appears, configure the component as required. For more information, see Process components. You can also copy or delete the component.
5. Select and configure output components based on your business scenario.
   i. Click the > icon before **Output**. Drag a component to the pipeline canvas on the left.
   ii. Right-click the component and select **Configure Attributes**. In the dialog box that appears, configure the component as required. For more information, see Output components. You can also copy or delete the component.
6. Click and hold Position 1 in an upstream component and drag it to Position 2 in a downstream component to draw a directed line and establish a relationship between the

components.



The following table describes the relationships that you can establish between components.

Component type	Description
Input	Note the following items when you use an input component: ○ An input component does not support upstream nodes. ○ The downstream nodes of an input component can be conversion components, output components, and process components. ○ If you connect an input component to multiple components such as output or conversion components in the downstream, you must select a data transmission mode for the input component.  ■ **Copy**: copies all the data of the upstream node to each downstream node. ■ **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data shares to the downstream nodes in turn. The sum of the data of all the downstream nodes is equal to that of the upstream node.
Output	An output component does not support downstream nodes.

Component type	Description
Process	Note the following items when you use a process component: ○ A process component can be used as any node except the first node and the last node in an offline migration pipeline. ○ If you connect a process component to multiple components such as conversion, output, or process components in the downstream, you must select a data transmission mode for the process component. ○ If you select a Conditional Distribution component as a process component, you must specify the conditions for data distribution when you connect it to downstream nodes.  ■ If you select **Conditioned Result Is true**, the Conditional Distribution component transmits data to downstream nodes only when the specified conditions are met. ■ If you select **Conditioned Result Is false**, the Conditional Distribution component transmits data to downstream nodes only when the specified conditions are not met.
Conversion	If you connect a conversion component to multiple components such as conversion, output, or process components in the downstream, you must select a data transmission mode for the conversion component.

You can perform the preceding steps to configure components for an offline migration pipeline on the pipeline canvas. A pipeline canvas allows you to configure multiple pipelines at the same time. You can also right-click a blank area on the pipeline canvas and perform the operations that are described in the following table.

Operation	Description
Copy	Copy the existing components that you select on the pipeline canvas.
Paste	Paste the copied components on the pipeline canvas.
Delete	Delete the components that you select from the pipeline canvas.
Circle	Select one or more components.
Select All	Select all components on the pipeline canvas.

7. After you configure components for the offline migration pipeline on the pipeline canvas, click the icon in the upper-right corner to save the pipeline. Then, the pipeline enters the **Draft** state.
8. Click the icon before **Pending** in the upper-left corner. If variables such as bizdate are configured in the components, set the parameters as required and click **OK** to run the pipeline.

Step 3: Configure the scheduling policy

1. On the pipeline configuration tab, click **Scheduling Configuration** in the top navigation bar. In the Scheduling Configuration pane, set the parameters as required.
   i. Set the parameters in the **Basic Information** section.Dataphin automatically generates the node name, node ID, node type, and owner. You cannot modify these parameters.

Parameter	Description
Description	The description of the scheduling policy.
Priority	The priority based on which the task corresponding to the offline migration pipeline is scheduled. Valid values: ■ Lowest Priority ■ Low Priority ■ Medium Priority ■ High Priority ■ Highest Priority
Parameters	The specified values for the parameters that are used in the code of the task. Dataphin allows you to specify the parameters that are used in the code of a task. The specified values are used when the task is run. You can click **Parameters and Descriptions** to know how to configure the parameters.

   ii. Set the parameters in the **Scheduling Configuration** section.

Parameter	Description

Parameter	Description
Schedule Mode	The scheduling mode of the task corresponding to the offline migration pipeline. Valid values: ■ **Normal**: runs the task based on the specified recurrence. By default, this option is selected. ■ **Dry-run**: runs the task based on the specified recurrence. However, the scheduling system does not run the task but returns a success response. ■ **Pause Scheduling**: runs the task based on the specified recurrence. However, the scheduling system does not run the task but returns a failure response. You can select this check box if you want to suspend a task and run it later.
Recurrence	The recurrence of the task. Valid values: ■ **Day**: automatically runs the task once per day. When you create a recurring task, the task is set to be run at 00:00 every day by default. You can also click the icon to specify a point in time for the task to be run as needed. ■ **Week**: automatically runs the task at a specified point in time on specified days of each week. You can also click the icon to specify a point in time for the task to be run as needed. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not run the instance or consume resources but returns a success response. Assume that you set Recurrence to Week and specify that the task is run every Monday and Tuesday. The scheduling system generates and runs instances every Monday and Tuesday. Every Wednesday, Thursday, Friday, Saturday, and Sunday, the scheduling system generates instances and returns success responses without running the instances. ■ **Month**: automatically runs the task at a specified point in time on specified days of each month. You can also click the icon to specify a point in time for the task to be run as needed. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not run the instance or consume resources but returns a success response. Assume that you set Recurrence to Month and specify that the task is run on the seventh day of each month. The scheduling system generates and runs an instance on the seventh day of each month. On the other days, the scheduling system generates instances and returns success responses without running the instances.

Parameter	Description
	■ **Hour**: automatically runs the task at a specified interval during a specified time period or at specified points in time every day. The scheduling system automatically generates instances for the task and runs the instances at the specified interval or points in time. Assume that you set Recurrence to Hour, select Time Period, and set the Start, End, and Interval parameters to 00:00, 23:59, and 1, respectively. The scheduling system automatically generates instances for the task and runs an instance every hour. ■ **Minute**: automatically runs the task at a specified interval during a specified time period every day. You can also click the icon to specify a point in time for the task to be run as needed. Assume that you set Recurrence to Minute and set the Start, End, and Interval parameters to 00:00, 23:59, and 05, respectively. The scheduling system automatically generates instances for the task and runs an instance every 5 minutes.
Depend on Previous Instance	Specifies whether the current task is run after the previous instance of another task or of the current task is run. If you select Depend on Previous Instance, you must further select Current Task or Select Task. ■ If you select **Current Task**, the current task is run after the previous instance of the current task is run. ■ If you select **Select Task**, enter a keyword in the search box that appears to search for and select one or more tasks to depend on.

iii. Set the parameters in the **Dependency** section.

Parameter	Description

Parameter	Description
Upstream Dependency	The upstream nodes on which the current node depends. To specify an upstream node, perform the following steps: a. Click **Create Upstream Dependency**. b. In the **Create Upstream Dependency** dialog box, search for the desired node based on the output name and select the node. **Note** Each node output name is globally unique in Dataphin. c. Click **OK**. To remove a node from the upstream node list, click the icon in the **Actions** column.
Current Node	The output name for the current node. You can set multiple output names for a node to be used for the dependency configuration of other nodes. To set an output name, perform the following steps: a. Click **Add**. b. In the **Add Output Task Nodes for Current Task Node** dialog box, enter an output name. Observe uniform rules when you set each output name for the current node. Generally, set an output name in the format of `Project name. Table name`. This helps other users find this node when they configure the upstream dependency for their nodes. c. Click **OK**. You can also perform the following operations on an existing output name: ■ To delete an output name, click the icon in the **Actions** column. ■ To view the downstream nodes after the current node is submitted or published, click the icon in the **Actions** column.

2. Click **OK**.

Step 4: Configure the channel control policy

1. On the pipeline configuration tab, click **Channel Configuration** in the top navigation bar. In the Channel Configuration pane, set the parameters as required and click **OK**.

Parameter	Description
Error Tolerance Configuration	The maximum number of errors allowed in the offline migration pipeline. After you specify a fault tolerance threshold, one of the following situations may occur when the corresponding task is being run: ○ The task fails if the number of cumulative errors in all components of the offline migration pipeline reaches the specified fault tolerance threshold. ○ The task continues if the number of cumulative errors in all components of the offline migration pipeline falls below the specified fault tolerance threshold.
Global Concurrency Configuration	The maximum number of components that can concurrently transmit data in the entire offline migration pipeline.
JVM Configuration	The Java virtual machine (JVM) parameters.

2. Click the icon in the upper-right corner to save the offline migration pipeline.
3. Click the icon in the upper-right corner to submit the offline migration pipeline.

> **Note** When you submit the pipeline, Dataphin checks whether you have the following permissions:
>
> ○ Read permission on the data source in each input component
>
> ○ Write permission on the data source in each output component

4. (Optional)Publish the offline migration pipeline.
   - If the current project is in Dev mode, publish the pipeline to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the pipeline after you submit it.

9.8.2.4.2. Edit an offline migration pipeline

An offline migration pipeline is used to migrate data. This topic describes how to edit an offline migration pipeline.

Prerequisites

Offline migration pipelines are created. For more information, see Create and configure an offline migration pipeline.

Context

To edit an offline migration pipeline, you must be the creator of the pipeline or the super administrator.

Procedure

1. Log on to the Dataphin console.

2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to edit and select **Change**. The configuration tab of the pipeline appears.
6. (Optional)Steal the lock of the offline migration pipeline.
   - If the pipeline is locked by yourself, you do not need to steal the lock.
   - If the pipeline is locked by another user, click the icon in the upper-right corner of the pipeline configuration tab to steal the lock.
7. On the pipeline configuration tab, configure the pipeline components as required. For more information, see Create and configure an offline migration pipeline.
8. Save and submit the offline migration pipeline.
   i. Click the icon in the upper-right corner of the pipeline configuration tab to save the pipeline.
   ii. Click the icon in the upper-right corner of the pipeline configuration tab to submit the pipeline.When you submit the pipeline, Dataphin checks whether you have the following permissions:
      - Read permission on the data source in each input component
      - Write permission on the data source in each output component
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
9. (Optional)Publish the offline migration pipeline.
   - If the current project is in Dev mode, publish the pipeline to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the pipeline after you submit it.

9.8.2.4.3. Rename an offline migration pipeline

An offline migration pipeline is used to migrate data. This topic describes how to rename an offline migration pipeline.

Prerequisites

Offline migration pipelines are created. For more information, see Create and configure an offline migration pipeline.

Procedure

1. Log on to the Dataphin console.

2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to rename and select **Rename**.
6. In the field that appears, enter a new name.
7. Press the Enter key.

9.8.2.4.4. Move an offline migration pipeline

An offline migration pipeline is used to migrate data. This topic describes how to move an offline migration pipeline to a specified directory.

Prerequisites

Offline migration pipelines are created. For more information, see Create and configure an offline migration pipeline.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to move and select **Move**.
6. In the **Move File** dialog box, set the Select Directory parameter to the destination directory.
7. Click **OK**.

9.8.2.4.5. View historical version information about an offline migration pipeline and download the script in a version of a pipeline

This topic describes how to view historical version information about an offline migration pipeline and download the script in a version of a pipeline.

Prerequisites

Offline migration pipelines are created. For more information, see Create and configure an offline migration pipeline.

View historical version information about an offline migration pipeline

1. Log on to the Dataphin console.
2. Go to the **Integrated** page.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
3. In the left-side navigation pane, click the pipeline that you want to view.
4. On the pipeline configuration tab, click **Versions** in the top navigation bar. If the pipeline is locked by another user, you must click the Lock icon in the upper-right corner of the pipeline configuration tab to steal the lock first.
5. In the **Node Version** pane, view the historical versions of the pipeline.
6. Click the icon in the **Actions** column of a version to view the version details.
7. On the **Pipeline Details** page, view the components, scheduling configuration, and fault tolerance configuration in the version.
8. Return to the pipeline configuration tab and click **OK** in the **Node Version** pane.

Download the script in a version of an offline migration pipeline

1. In the left-side navigation pane, click the pipeline that you want to view.
2. On the pipeline configuration tab, click **Versions** in the top navigation bar. If the pipeline is locked by another user, you must click the Lock icon in the upper-right corner of the pipeline configuration tab to steal the lock first.
3. In the **Node Version** pane, find a version and click the icon in the **Actions** column.

9.8.2.4.6. Unpublish and delete an offline migration pipeline

This topic describes how to unpublish, unpublish and delete, and delete offline migration pipelines.

Prerequisites

Offline migration pipelines are created. For more information, see Create and configure an offline migration pipeline.

Context

- Description on the status of an offline migration pipeline:
  - After you create and save a pipeline, it enters the **Draft** state.

- After you create and save a pipeline, it enters the **Draft** state.
  - After you submit a pipeline, it enters the **Submitted** state.
  - After you edit and save a pipeline in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a pipeline in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing offline migration pipelines:
  - You can unpublish only the pipelines in the **Developing** or **Submitted** state.
  - To unpublish a pipeline, you must be the creator of the pipeline or the super administrator.
- Limits on deleting offline migration pipelines:
  - To delete a pipeline, you must be the creator of the pipeline or the super administrator.
  - You can delete only the pipelines in the **Draft** state.
- Limits on unpublishing and deleting offline migration pipelines:
  - You can unpublish and delete only the pipelines in the **Developing** or **Submitted** state.
  - To unpublish and delete a pipeline, you must be the creator of the pipeline or the super administrator.

Unpublish an offline migration pipeline

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to unpublish and select **Unpublish**.
6. In the **Tip** dialog box, enter your comments.
7. Click **OK**.

Unpublish and delete an offline migration pipeline

1. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to unpublish and delete and select **Unpublish and Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

Delete an offline migration pipeline

1. In the left-side navigation pane, move the pointer over the icon next to the pipeline that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.8.2.5. Manage offline database migration tasks

9.8.2.5.1. Create and configure an offline database migration task

Dataphin allows you to manage offline database migration tasks. An offline database migration task consists of multiple offline migration pipelines that can migrate multiple tables from one database to another at a time. This topic describes how to create and configure an offline database migration task.

Context

An offline database migration task allows you to migrate multiple tables from one database to another at a time. This improves the configuration efficiency and reduces costs.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. On the **Integrated** page, open the **Create Database Migration** tab by using one of the following methods:
   - In the Script section, move the pointer over the icon next to Script and select **Offline Database Migration**.
   - In the left-side navigation pane, move the pointer over the icon next to the project name and select **Offline Database Migration**.
6. On the **Create Database Migration** tab, set the parameters as required.
   i. Set the parameters in the **Basic Information** section.

Parameter	Description
Script Name	The name of the offline database migration task. The name can be up to 64 characters in length and can contain letters, digits, and underscores (_).
Description	The description of the task.

ii. Set the parameters in the Data Source Configuration section.

Section	Parameter	Description
Synchronization Source	Data Source Type	The type of the data source from which data is read. Valid values: ■ MYSQL ■ ORACLE ■ SQL_SERVER
	Data Source	The name of the data source from which data is read. You can also click Create Data Source in the Data Source drop-down list to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Synchronization Target	Data Source Type	The type of the data source to which data is written. You can set this parameter only to MAX_COMPUTE.
	Data Source	The name of the data source to which data is written. You can also click Create Data Source in the Data Source drop-down list to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.

iii. Set the parameters in the Table Synchronization section.

GUI element	Description
Source Table	The source tables to be migrated.
Target MaxCompute Table	The destination tables to be generated in the specified MaxCompute project. After you select a source table in the Source Table column, a same table name appears in the Target MaxCompute Table column. MaxCompute automatically creates tables based on the table names in the Target MaxCompute Table column.

GUI element	Description
Configure Conversion	The rules for converting table names and field names. By default, the source tables and destination tables use the same names. This button allows you to modify the configuration of the destination tables. To modify the configuration of the destination tables, perform the following steps: a. Click **Configure Conversion**. b. In the **Edit Conversion Rule** dialog box, perform one or more of the following operations as required: ■ In the **Table Name Conversion** section, click **Create Rule** and set conversion rules for the names of the source and destination tables. ■ In the **Field Name Conversion** section, click **Create Rule** and set conversion rules for the field names in the source and destination tables. ■ In the **Table Name Prefix** section, enter a prefix for the names of the destination tables. ■ In the **Data Filtering** section, enter a filter condition for input fields, for example, `gmt_modified>=${bizdate}` . c. Click **Save and Execute**. The configured conversion rules apply to the destination tables that are listed in the **Target MaxCompute Table** column.

iv. Set the parameters in the Synchronization Configuration section.

Parameter	Description
Synchronization Type	The synchronization mode of the task. Valid values: **Everyday, Full** and **Everyday, Incremental**.
Synchronization Concurrency	The synchronization concurrency mode of the task. Valid values: **Everyday, Full** and **Batch Upload**.

7. Click **Generate Pipeline**.View the pipeline generation result.
8. (Optional)Publish the task.
   - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the task after you submit it.

9.8.2.5.2. Manage offline migration pipelines

After you create an offline database migration task, multiple offline migration pipelines are accordingly generated. This topic describes how to manage an offline migration pipeline that is generated in an offline database migration task.

Prerequisites

Offline database migration tasks are created. For more information, see Create and configure an offline database migration task.

Context

- To edit an offline migration pipeline, you must be the creator of the pipeline or the super administrator.
- Description on the status of an offline migration pipeline:
  - After you create and save a pipeline, it enters the **Draft** state.
  - After you submit a pipeline, it enters the **Submitted** state.
  - After you edit and save a pipeline in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a pipeline in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing offline migration pipelines:
  - You can unpublish only the pipelines in the **Developing** or **Submitted** state.
  - To unpublish a pipeline, you must be the creator of the pipeline or the super administrator.
- Limits on deleting offline migration pipelines:
  - To delete a pipeline, you must be the creator of the pipeline or the super administrator.
  - You can delete only the pipelines in the **Draft** state.
- Limits on unpublishing and deleting offline migration pipelines:
  - You can unpublish and delete only the pipelines in the **Developing** or **Submitted** state.
  - To unpublish and delete a pipeline, you must be the creator of the pipeline or the super administrator.

Edit an offline migration pipeline

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.In the left-side navigation pane of the **Integrated** page:
   - The icon indicates a folder where offline migration pipelines reside.
   - The icon indicates an offline database migration task.

- The icon indicates an offline migration pipeline.

6. Move the pointer over the icon next to the pipeline that you want to edit and select **Change**.
7. (Optional)Steal the lock of the offline migration pipeline.
   - If the pipeline is locked by yourself, you do not need to steal the lock.
   - If the pipeline is locked by another user, click the icon in the upper-right corner of the pipeline configuration tab to steal the lock.
8. On the pipeline configuration tab, configure the pipeline components as required. For more information, see Edit an offline migration pipeline.
9. Save, submit, and then publish the offline migration pipeline.
   i. After you edit the pipeline components, click the icon in the upper-right corner of the pipeline configuration tab to save the pipeline.
   ii. Click the icon in the upper-right corner of the pipeline configuration tab to submit the pipeline.When you submit the pipeline, Dataphin checks whether you have the following permissions:
      - Read permission on the data source in each input component
      - Write permission on the data source in each output component
   iii. Optional. If the current project is in Dev mode, publish the offline migration pipeline to the corresponding project in Prod mode. For more information, see Publishing management.
      - After you publish the pipeline to the project in Prod mode, the pipeline can be used for scheduling.
      - If you submit the pipeline in a project in Basic mode, the pipeline can immediately be used for scheduling.

Rename an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. In the left-side navigation pane, move the pointer over next to the target pipeline and select **Rename**.
3. In the field that appears, enter a new name.
4. Press Enter.

Move an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. In the left-side navigation pane, move the pointer over next to the target pipeline and

select **Move**.

3. In the **Move Folder** dialog box, set the Select Directory parameter to the destination directory.
4. Click **OK**.

View historical version information about an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. In the left-side navigation pane, click the target pipeline.
3. On the pipeline configuration tab, click **Versions** in the top navigation bar. If the pipeline is locked by another user, you must click the Lock icon in the upper-right corner of the pipeline configuration tab to steal the lock first.
4. In the **Node Version** pane, view the historical versions of the pipeline.
5. Click in the **Actions** column of a version to view the version details.
6. On the **Pipeline Details** page, view the components, scheduling configuration, and error tolerance configuration in the version.
7. Return to the pipeline configuration tab and click **OK** in the **Node Version** pane.

Download the script in a version of an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. Click the pipeline that you want to view.
3. On the pipeline configuration tab, click **Versions** in the top navigation bar. If the pipeline is locked by another user, you must click the Lock icon in the upper-right corner of the pipeline configuration tab to steal the lock first.
4. In the **Node Version** pane, find the target version and click in the **Actions** column.

Unpublish an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. In the left-side navigation pane, move the pointer over next to the target pipeline and select **Unpublish**.
3. In the **Tip** dialog box, enter your comments.
4. Click **OK**.

Unpublish and delete an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage

and click the icon before the icon to show the list of pipelines that are generated in the task.

2. Move the pointer over the icon next to the pipeline that you want to unpublish and delete and select **Unpublish and Delete**.
3. In the **Tip** dialog box, enter your comments.
4. Click **OK**.

Delete an offline migration pipeline

1. On the **Integrated** page, find the offline database migration task that you want to manage and click the icon before the icon to show the list of pipelines that are generated in the task.
2. Move the pointer over the icon next to the pipeline that you want to delete and select **Delete**.
3. In the **Tip** dialog box, enter your comments.
4. Click **OK**.

9.8.2.5.3. Rename an offline database migration task

Dataphin allows you to manage offline database migration tasks. An offline database migration task consists of multiple offline migration pipelines that can migrate multiple tables from one database to another at a time. This topic describes how to rename an offline database migration task.

Prerequisites

Offline database migration tasks are created. For more information, see Create and configure an offline database migration task.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the offline database migration task that you want to rename.
6. Select **Rename**.
7. In the field that appears, enter a new name.
8. Press the Enter key.

9.8.2.5.4. Delete an offline database migration task

Dataphin allows you to manage offline database migration tasks. An offline database migration task consists of multiple offline migration pipelines that can migrate multiple tables from one database to another at a time. This topic describes how to delete an offline database migration task.

Prerequisites

Offline database migration tasks are created. For more information, see Create and configure an offline database migration task.

Context

If you delete an offline database migration task, all the offline migration pipelines that are generated in the task will also be deleted. Proceed with caution.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
5. In the left-side navigation pane, move the pointer over the icon next to the offline database migration task that you want to delete.
6. Select **Delete**.

9.8.2.6. Input components

9.8.2.6.1. Manage MySQL input components

This topic describes how to configure, copy, and delete a MySQL input component, and how to select a data sending mode for a MySQL input component.

Context

A MySQL input component is used to import data from a MySQL database. Then, you can write the data to a database, such as another MySQL database, for data consumption.

Configure a MySQL input component

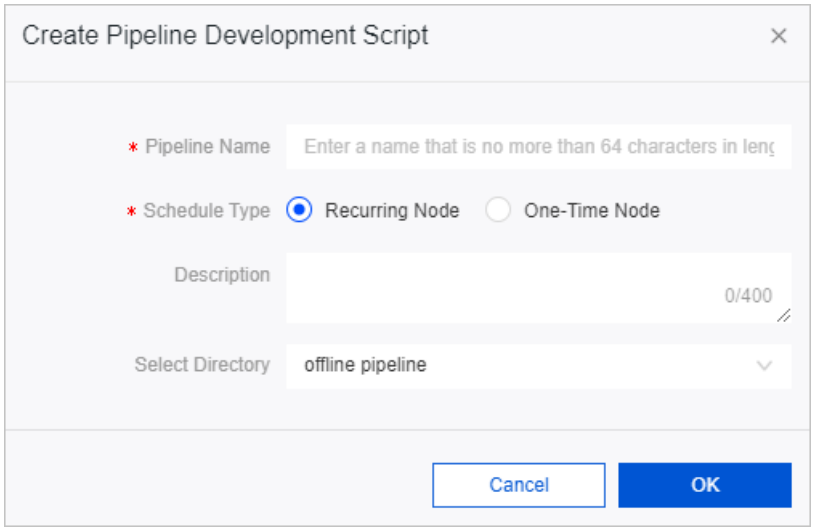
1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.

iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Input**.
5. Drag a MySQL component to the pipeline canvas on the left.
6. Right-click the MySQL component and select **Configure Attributes**.
7. In the **MySQL Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Data Source	The data source of the component. Select a data source that has been connected to Dataphin. The data source must meet the following requirements: ○ The data source is of the MySQL type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Source Table	The quantity of source tables. Valid values: **Single Table** and **Multiple Tables**.
Table	The source tables to be migrated. If you set the Source Table parameter to **Multiple Tables**, perform the following steps to add tables: i. Enter an expression in the Table field. Dataphin supports an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. ii. Click the icon. iii. In the **Confirm Match Details** dialog box, select the matching tables. iv. Click **OK**. **Note** The source tables that you select must have the same schema.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key.
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.

Parameter	Description
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy a MySQL input component

1. Right-click a MySQL input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a MySQL input component

1. Right-click a MySQL input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a MySQL input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to downstream nodes.

1. Right-click a MySQL input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are provided:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data shares to the downstream nodes in turn. The sum of the data of all the downstream nodes is equal to that of the upstream node.

9.8.2.6.2. Manage Maxcompute input components

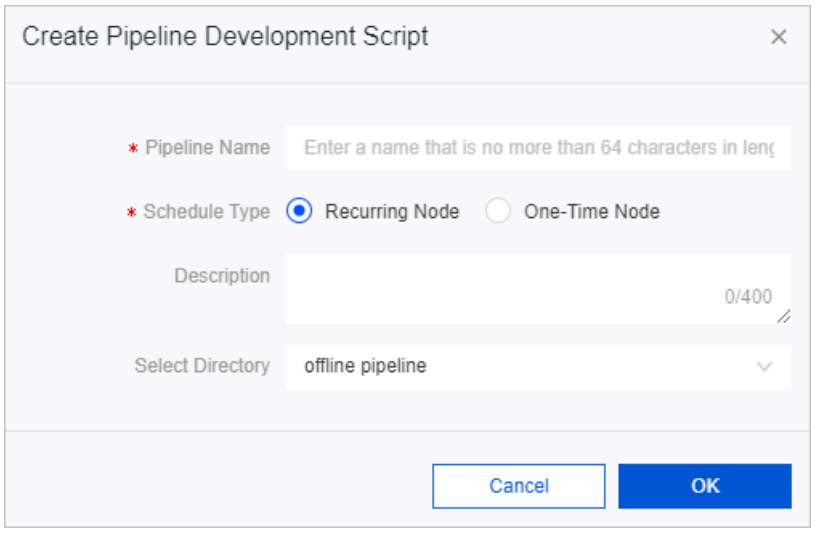
This topic describes how to configure, copy, and delete a Maxcompute input component, and how to select a data sending mode for a Maxcompute input component.

Context

A Maxcompute input component is used to import data from a MaxCompute project. Then, you can write the data to a database, such as a MySQL database, for data consumption.

Configure a Maxcompute input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.
   v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

   vi. Click **OK**.
3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Input**.

5. Drag a Maxcompute component to the pipeline canvas on the left.
6. Right-click the Maxcompute component and select **Configure Attributes**.
7. In the **Maxcompute Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Data Source	The data source of the component. Select a data source that has been connected to Dataphin. The data source must meet the following requirements: ○ The data source is of the MaxCompute type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The source table to be migrated. If the table you selected is a partitioned table, enter the partition information, for example, `ds=${bizdate}`.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy a Maxcompute input component

1. Right-click a Maxcompute input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Maxcompute input component

1. Right-click a Maxcompute input component and select **Delete**.

2. In the message that appears, click **OK**.

Select a data sending mode for a Maxcompute input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to downstream nodes.

1. Right-click a Maxcompute input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are provided:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data shares to the downstream nodes in turn. The sum of the data of all the downstream nodes is equal to that of the upstream node.

9.8.2.6.3. Manage Vertica input components

This topic describes how to configure, copy, and delete a Vertica input component, and how to select a data sending mode for a Vertica input component.

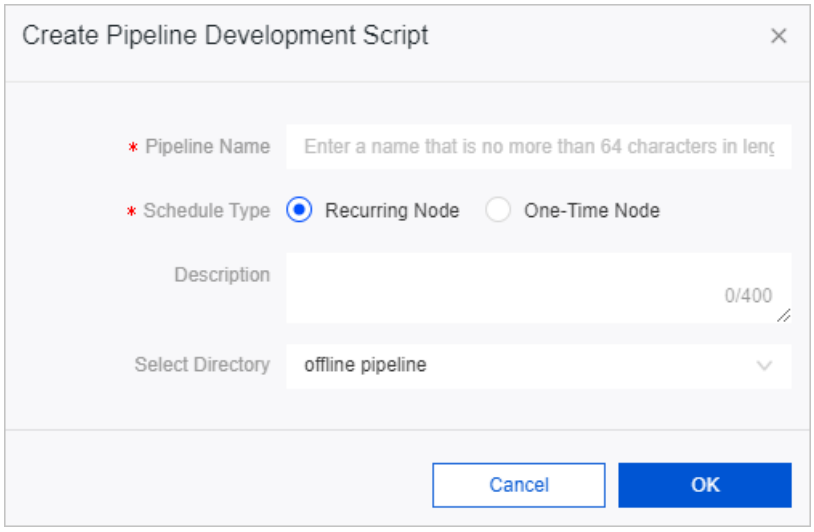
Context

A Vertica input component is used to import data from a Vertica database to Dataphin for integration and reprocessing.

Configure a Vertica input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a Vertica component to the pipeline canvas on the left.
6. Right-click the Vertica component and select **Configure Attributes**.
7. In the **Vertica Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been connected to Dataphin. The data source must meet the following requirements: ○ The data source is of the Vertica type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a Vertica data source.
Table	The source table to be migrated.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key.
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy a Vertica input component

1. Right-click a Vertica input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Vertica input component

1. Right-click a Vertica input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a Vertica input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to downstream nodes.

1. Right-click a Vertica input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are provided:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data shares to the downstream nodes in turn. The sum of the data of all the downstream nodes is equal to that of the upstream node.

9.8.2.6.4. Manage DRDS input components

This topic describes how to configure, copy, and delete a DRDS input component, and how to select a data sending mode for a DRDS input component.

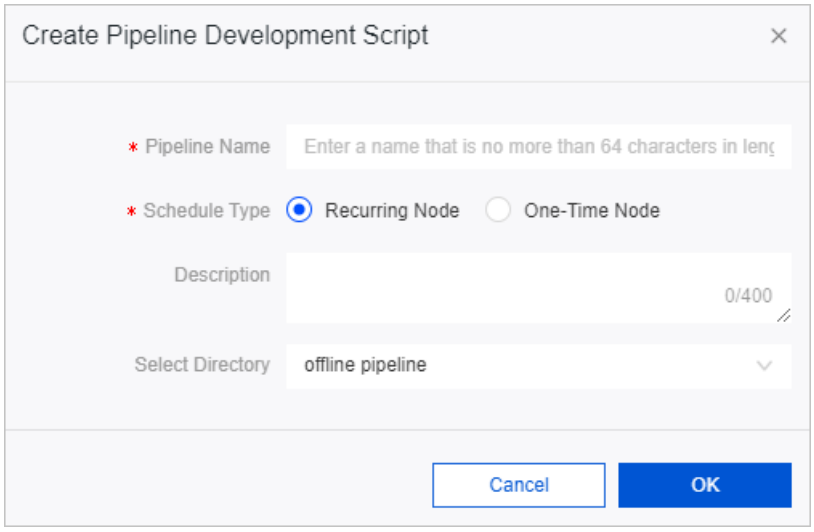
Context

A DRDS input component is used to import data from a PolarDB-X database to Dataphin for integration and reprocessing.

Configure a DRDS input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a DRDS component to the pipeline canvas on the left.
6. Right-click the DRDS component and select **Configure Attributes**.
7. In the **DRDS Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the PolarDB-X type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to Data Source to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The source table.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key.
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The Input Filtering parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the Actions column to delete a field. ○ Click Manage Fields. In the Manage Fields dialog box, perform the following operations: ■ Select one or more fields in the Unselected Input Fields section and click the icon to move the fields to the Selected Input Fields section. ■ Select one or more fields in the Selected Input Fields section and click the icon to move the fields to the Unselected Input Fields section.

8. Click OK.

Copy a DRDS input component

1. Right-click a DRDS input component and select Copy.
2. Right-click a blank area on the pipeline canvas and select Paste.

Delete a DRDS input component

1. Right-click a DRDS input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a DRDS input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a DRDS input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.5. Manage PostgreSQL input components

This topic describes how to configure, copy, and delete a PostgreSQL input component, and how to select a data sending mode for a PostgreSQL input component.

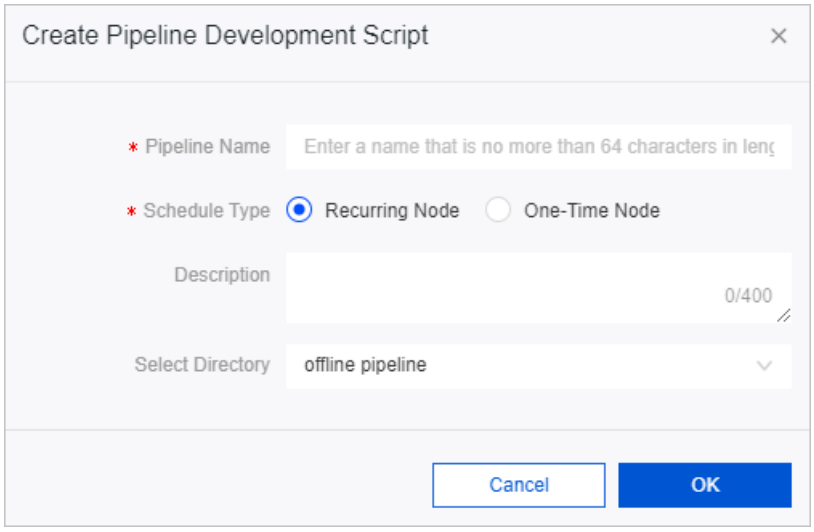
Context

A PostgreSQL input component is used to import data from a PostgreSQL database to Dataphin for integration and reprocessing.

Configure a PostgreSQL input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a PostgreSQL component to the pipeline canvas on the left.
6. Right-click the PostgreSQL component and select **Configure Attributes**.
7. In the **PostgreSQL Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the PostgreSQL type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to Data Source to go to the Data Sources page and create a data source. For more information, see Create a PostgreSQL data source.
Source Table	The quantity of source tables. Valid values: Single Table and Multiple Tables. Note If you select Multiple Tables, you must specify multiple source tables that have the same schema.
Table	The one or more source tables. ○ If you set Source Table to Single Table, click the icon and select a source table from the Table drop-down list. ○ If you set Source Table to Multiple Tables, perform the following steps to specify the source tables: a. Enter an expression in the Table field. Dataphin allows you to enter an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. b. Click the icon. c. In the Confirm Match Details dialog box, select the tables that match the expression you entered. d. Click OK.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. If you specify multiple source tables, you need to specify only one shard key because the source tables have the same schema.

Parameter	Description
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, view the unselected and selected input fields and perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy a PostgreSQL input component

1. Right-click a PostgreSQL input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a PostgreSQL input component

1. Right-click a PostgreSQL input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a PostgreSQL input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a PostgreSQL input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.6. Manage SQL Server input components

This topic describes how to configure, copy, and delete an SQL Server input component, and how to select a data sending mode for an SQL Server input component.

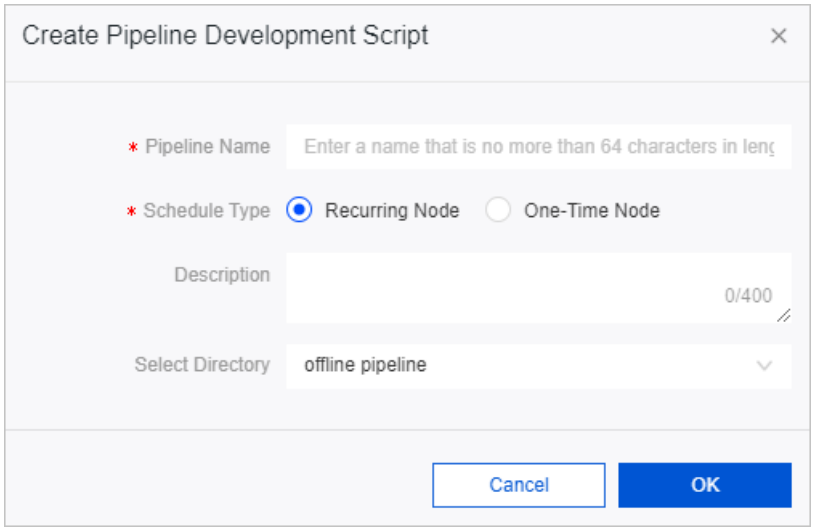
Context

An SQL Server input component is used to import data from an SQL Server database to Dataphin for integration and reprocessing.

Configure an SQL Server input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an SQL Server component to the pipeline canvas on the left.
6. Right-click the SQL Server component and select **Configure Attributes**.
7. In the **SQL Server Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the SQL Server type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Source Table	The quantity of source tables. Valid values: **Single Table** and **Multiple Tables**. **Note** If you select **Multiple Tables**, you must specify multiple source tables that have the same schema.
Table	The one or more source tables. ○ If you set **Source Table** to **Single Table**, click the icon and select a source table from the **Table** drop-down list. ○ If you set **Source Table** to **Multiple Tables**, perform the following steps to specify the source tables: a. Enter an expression in the Table field. Dataphin allows you to enter an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. b. Click the icon. c. In the **Confirm Match Details** dialog box, select the tables that match the expression you entered. d. Click **OK**.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. If you specify multiple source tables, you need to specify only one shard key because the source tables have the same schema.

Parameter	Description
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy an SQL Server input component

1. Right-click an SQL Server input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an SQL Server input component

1. Right-click an SQL Server input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an SQL Server input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an SQL Server input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.7. Manage ORACLE input components

This topic describes how to configure, copy, and delete an ORACLE input component, and how to select a data sending mode for an ORACLE input component.

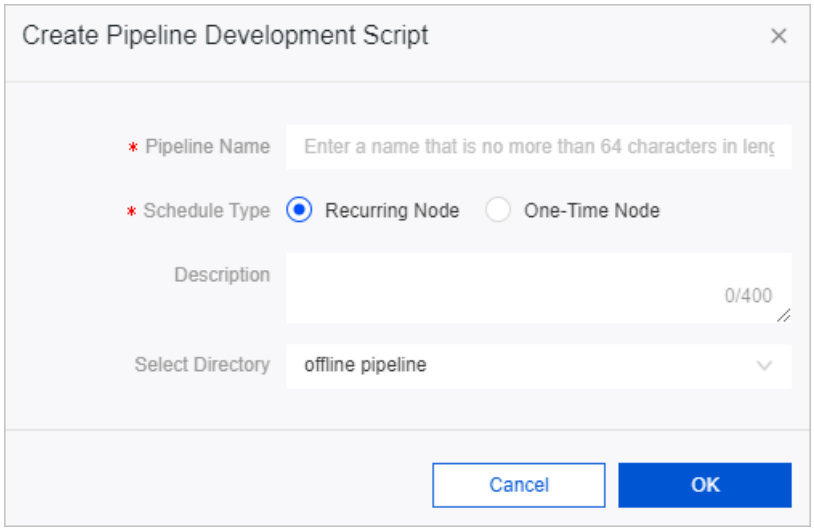
Context

An ORACLE input component is used to import data from an Oracle database to Dataphin for integration and reprocessing.

Configure an ORACLE input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an Oracle component to the pipeline canvas on the left.
6. Right-click the Oracle component and select **Configure Attributes**.
7. In the **Oracle Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the Oracle type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an Oracle data source.
Source Table	The quantity of source tables. Valid values: **Single Table** and **Multiple Tables**. **Note** If you select **Multiple Tables**, you must specify multiple source tables that have the same schema.
Table	The one or more source tables. ○ If you set **Source Table** to **Single Table**, click the icon and select a source table from the **Table** drop-down list. ○ If you set **Source Table** to **Multiple Tables**, perform the following steps to specify the source tables: a. Enter an expression in the Table field. Dataphin allows you to enter an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. b. Click the icon. c. In the **Confirm Match Details** dialog box, select the tables that match the expression you entered. d. Click **OK**.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. If you specify multiple source tables, you need to specify only one shard key because the source tables have the same schema.
Encoding	The encoding format of the source table. Valid values: **UTF-8**, **GBK**, and **ISO-8859-1**.

Parameter	Description
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy an ORACLE input component

1. Right-click an ORACLE input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ORACLE input component

1. Right-click an ORACLE input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an ORACLE input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an ORACLE input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.8. Manage FTP input components

This topic describes how to configure, copy, and delete an FTP input component, and how to select a data sending mode for an FTP input component.

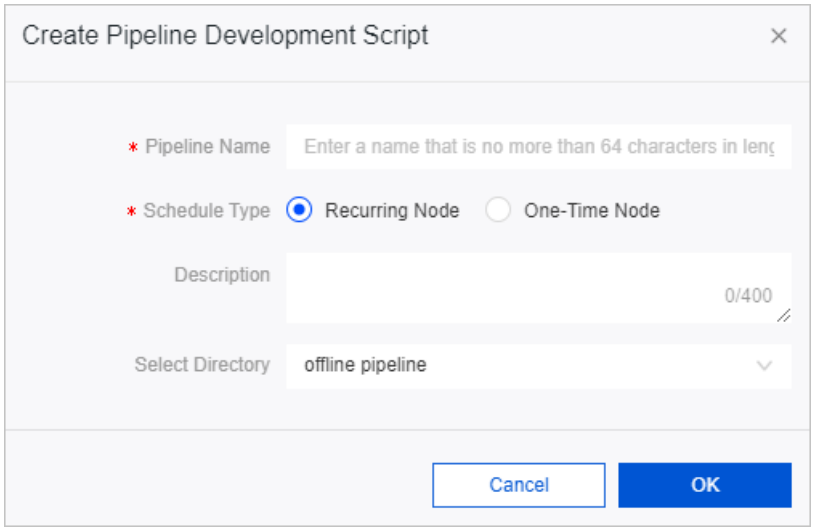
Context

An FTP input component is used to import data from an FTP server to Dataphin for integration and reprocessing.

Configure an FTP input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an FTP component to the pipeline canvas on the left.
6. Right-click the FTP component and select **Configure Attributes**.
7. In the **FTP Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the FTP or SFTP type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the  icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an FTP data source.
File Path	The path of the source file.
File Type	The type of the source file. Valid values: **TexT** and **CSV**.
File Encoding	The encoding format of the source file. Valid values: **UTF-8** and **GBK**.
Field Delimiter	The delimiter that is used to separate fields in the source file. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Compression Format	The compression format of the source file. Valid values: ○ **zip** ○ **gzip** ○ **bzip2**
First Line Type	The content type of the first line of the source file. ○ If you select **Data Content**, you must configure output fields by using one of the following methods: ■ Click **Create Output Field**. Set the **Source Serial Number**, **Field**, and **Type** parameters as required. ■ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. ○ If you select **Field Name**, the output fields are automatically parsed based on the fields in the first line of the source file.

Parameter	Description
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "index": 0, "name": "user_id", "type": "String" }, { "index": 1, "name": "user_name", "type": "String" }]` In the example, index, name, and type specify the sequence number, name, and data type of the output field, respectively. ○ Click **Create Output Field**. Set the **Source Serial Number**, **Field**, and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy an FTP input component

1. Right-click an FTP input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an FTP input component

1. Right-click an FTP input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an FTP input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an FTP input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:

- **Copy**: copies all the data of the upstream node to each downstream node.
- **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.9. Manage HDFS input components

This topic describes how to configure, copy, and delete an HDFS input component, and how to select a data sending mode for an HDFS input component.

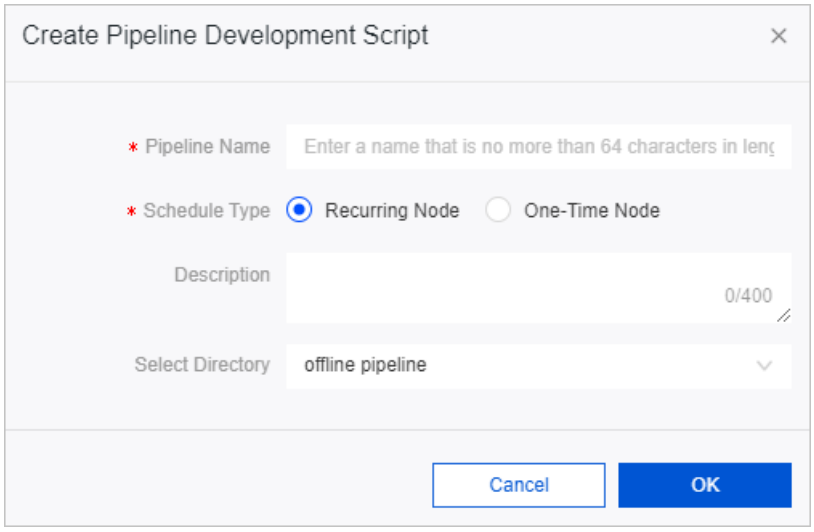
Context

An HDFS input component is used to import data from an HDFS to Dataphin for integration and reprocessing.

Configure an HDFS input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an HDFS component to the pipeline canvas on the left.
6. Right-click the HDFS component and select **Configure Attributes**.
7. In the **HDFS Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the HDFS type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an HDFS data source.
File Path	The path of the source file.
File Type	The type of the source file. Valid values: **TexT**, **CSV**, **ORC**, **RC**, and **Sequence**.
File Encoding	The encoding format of the source file. Valid values: **UTF-8** and **GBK**.
Field Delimiter	The delimiter that is used to separate fields in the source file. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Compression Format	The compression format of the source file. Valid values: ○ **zip** ○ **gzip** ○ **bzip2**

Parameter	Description
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "index": 0, "name": "user_id", "type": "String" }, { "index": 1, "name": "user_name", "type": "String" }]` In the example, index, name, and type specify the sequence number, name, and data type of the output field, respectively. ○ Click **Create Output Field**. Set the **Source Serial Number**, **Field**, and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy an HDFS input component

1. Right-click an HDFS input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an HDFS input component

1. Right-click an HDFS input component and select **Delete**.

2. In the message that appears, click **OK**.

Select a data sending mode for an HDFS input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an HDFS input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.10. Manage Hive input components

This topic describes how to configure, copy, and delete a Hive input component, and how to select a data sending mode for a Hive input component.

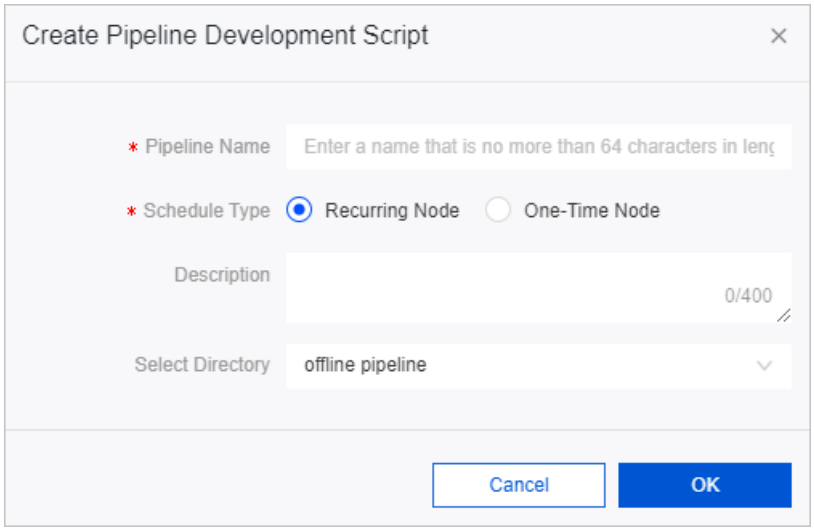
Context

A Hive input component is used to import data from a Hive database to Dataphin for integration and reprocessing.

Configure a Hive input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a Hive component to the pipeline canvas on the left.
6. Right-click the Hive component and select **Configure Attributes**.
7. In the **Hive Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the Hive type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The source table.
File Encoding	The encoding format of the source table. Valid values: **UTF-8** and **GBK**.
Compression Format	The format that is used to compress the source file. Valid values: ○ **zip** ○ **gzip** ○ **bzip2**
Field Delimiter	The delimiter that is used to separate fields in the source table. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Output Fields	The output fields to be generated based on the input configuration. You can click **Manage Fields** and perform the following operations in the **Manage Fields** dialog box: ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. You can also click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a Hive input component

1. Right-click a Hive input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Hive input component

1. Right-click a Hive input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a Hive input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a Hive input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.11. Manage Hbase input components

This topic describes how to configure, copy, and delete an Hbase input component, and how to select a data sending mode for an Hbase input component.

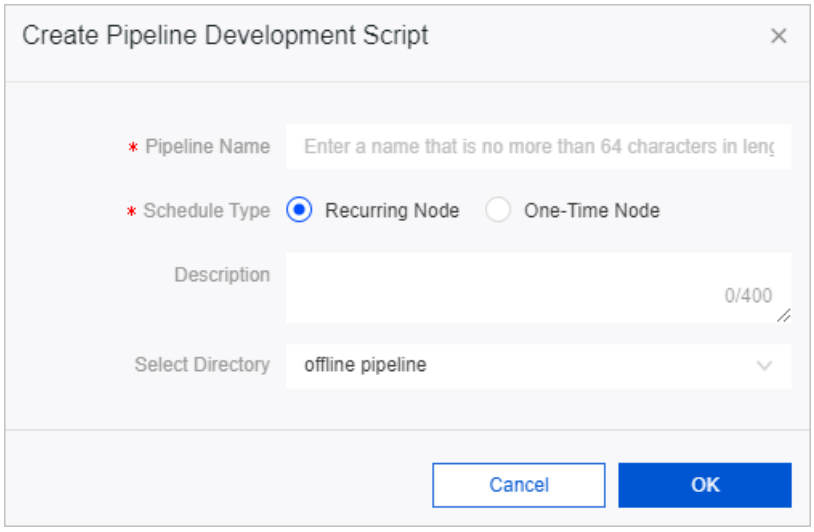
Context

An Hbase input component is used to import data from an HBase database to Dataphin for integration and reprocessing.

Configure an Hbase input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an Hbase component to the pipeline canvas on the left.
6. Right-click the Hbase component and select **Configure Attributes**.
7. In the **HBase Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the HBase type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an HBase data source.
Table	The source table.
Mode	The mode of the source table. Valid values: **Normal Mode** and **Multiple Versions**. If you select **Multiple Versions**, you must set the Version parameter to specify the versions of data to read. If you set the Version parameter to -1, all versions of data will be read.
File Encoding	The encoding format of the source table. Valid values: **UTF-8** and **GBK**.
Input Filtering	The filter condition for input fields. For example, you can enter `"startRowkey":"20190101000000"`.

Parameter	Description
Output Fields	The output fields to be generated based on the input configuration. You can create output fields by using one of the following methods: ○ Click **Create Output Field**. Set the **Column Family**, **Field**, and **Type** parameters as required. ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format and click **OK**. Example: `[{ "index": 0, "name": "user_id", "type": "String" }, { "index": 1, "name": "user_name", "type": "String" }]` In the example, index, name, and type specify the sequence number, name, and data type of the output field, respectively. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click the icon in the **Actions** column to edit a field.

8. Click **OK**.

Copy an Hbase input component

1. Right-click an Hbase input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an Hbase input component

1. Right-click an Hbase input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an Hbase input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an Hbase input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.

- **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.12. Manage MongoDB input components

This topic describes how to configure, copy, and delete a MongoDB input component, and how to select a data sending mode for a MongoDB input component.

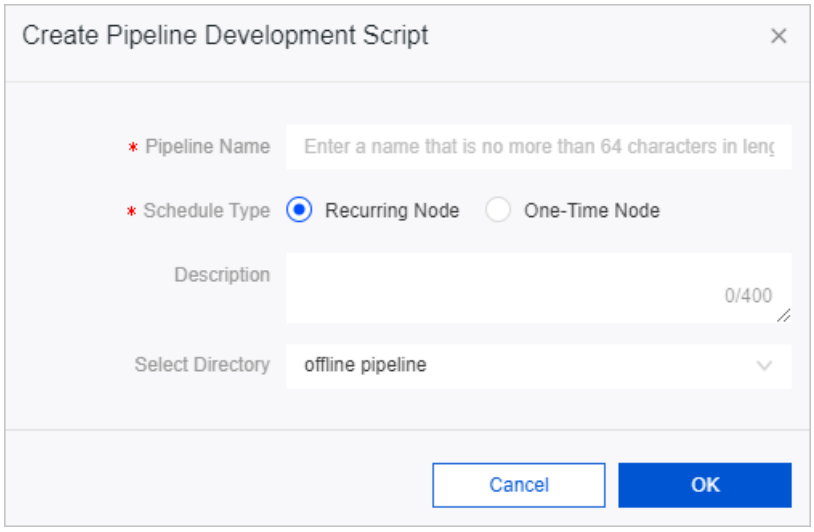
Context

A MongoDB input component is used to import data from a MongoDB database to Dataphin for integration and reprocessing.

Configure a MongoDB input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a MongoDB component to the pipeline canvas on the left.
6. Right-click the MongoDB component and select **Configure Attributes**.
7. In the **MongoDB Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the MongoDB type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The source table.
Input Filtering	The filter condition for input fields. For example, you can enter *ds=${biz date}*. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format and click **OK**. Example: `[{ "name":"cf1:a", "type":"String" },` `{ "name":"cf1:b", "type":"String" }]` ○ Click **Create Output Field**. Set the **Field** and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a MongoDB input component

1. Right-click a MongoDB input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a MongoDB input component

1. Right-click a MongoDB input component and select **Delete**.

2. In the message that appears, click **OK**.

Select a data sending mode for a MongoDB input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a MongoDB input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.13. Manage LogicalTable input components

This topic describes how to configure, copy, and delete a LogicalTable input component, and how to select a data sending mode for a LogicalTable input component.

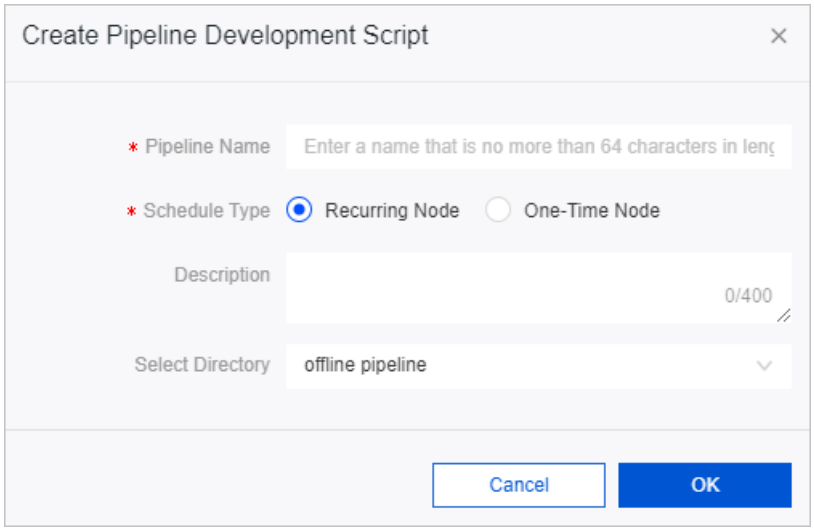
Context

A LogicalTable input component is used to import data from a logical table created in Dataphin to a database for integration and reprocessing.

Configure a LogicalTable input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a LogicalTable component to the pipeline canvas on the left.
6. Right-click the LogicalTable component and select **Configure Attributes**.
7. In the **Logical Table Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Business Units	The business unit where the logical table resides.

Parameter	Description
Logical Table Type	The type of the logical table. Valid values: ○ **Logical Aggregate Table** ○ **Logical Dimension Table** ○ **Logical Fact Table**
Logical Tables	The logical table. The system displays the logical tables on which your account has the read-through permission. You can also click the icon to go to the Develop page and create a logical table.
Logical Table Output Mode	The output mode of the logical table. Valid values: ○ **Without Association**: The output fields to be generated are the fields of the selected logical table. ○ **With Association**: The output fields to be generated are the fields of the selected logical table and associated table.
Output Fields	The output fields to be generated. You can perform the following steps to preview the output fields: i. Click **Manage Fields**. ii. Select one or more fields in the **All Fields** section. The output fields to be generated appear in the Output Field Preview section. You can also click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a LogicalTable input component

1. Right-click a LogicalTable input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a LogicalTable input component

1. Right-click a LogicalTable input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a LogicalTable input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a LogicalTable input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream

node.

9.8.2.6.14. Manage ADB for MySQL3.0 input components

This topic describes how to configure, copy, and delete an ADB for MySQL3.0 input component, and how to select a data sending mode for an ADB for MySQL3.0 input component.

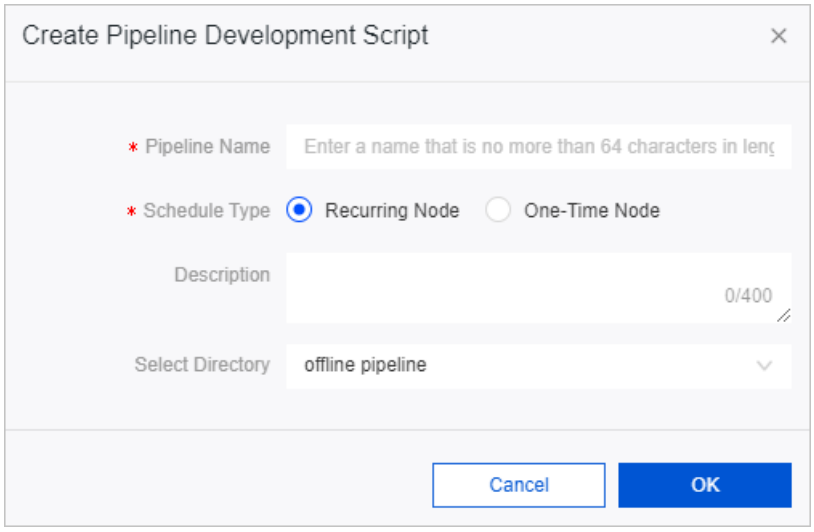
Context

An ADB for MySQL3.0 input component is used to import data from an AnalyticDB for MySQL 3.0 database to Dataphin for integration and reprocessing.

Configure an ADB for MySQL3.0 input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an **ADB for MySQL3.0** component to the pipeline canvas on the left.
6. Right-click the **ADB for MySQL3.0** component and select **Configure Attributes**.
7. In the **ADB for MySQL 3.0 Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the input component. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an AnalyticDB for MySQL V3.0 data source.
Source Table	The quantity of source tables. Valid values: **Single Table** and **Multiple Tables**. **Note** If you select **Multiple Tables**, you must specify multiple source tables that have the same schema.
Table	The one or more source tables. Specify one or more source tables based on the **Source Table** parameter. ○ If you set **Source Table** to **Single Table**, click the icon and select a source table from the **Table** drop-down list. ○ If you set **Source Table** to **Multiple Tables**, perform the following steps to specify the source tables: a. Enter an expression in the Table field. Dataphin allows you to enter an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. b. Click the icon. c. In the **Confirm Match Details** dialog box, select the tables that match the expression you entered. d. Click **OK**.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. If you specify multiple source tables, you need to specify only one shard key because the source tables have the same schema.

Parameter	Description
Input Filtering	The filter condition for input fields. For example, you can enter *ds=${biz date}*. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy an ADB for MySQL3.0 input component

1. Right-click an ADB for MySQL3.0 input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ADB for MySQL3.0 input component

1. Right-click an ADB for MySQL3.0 input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an ADB for MySQL3.0 input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an ADB for MySQL3.0 input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.15. Manage ADB for PostgreSQL input components

This topic describes how to configure, copy, and delete an ADB for PostgreSQL input component, and how to select a data sending mode for an ADB for PostgreSQL input component.

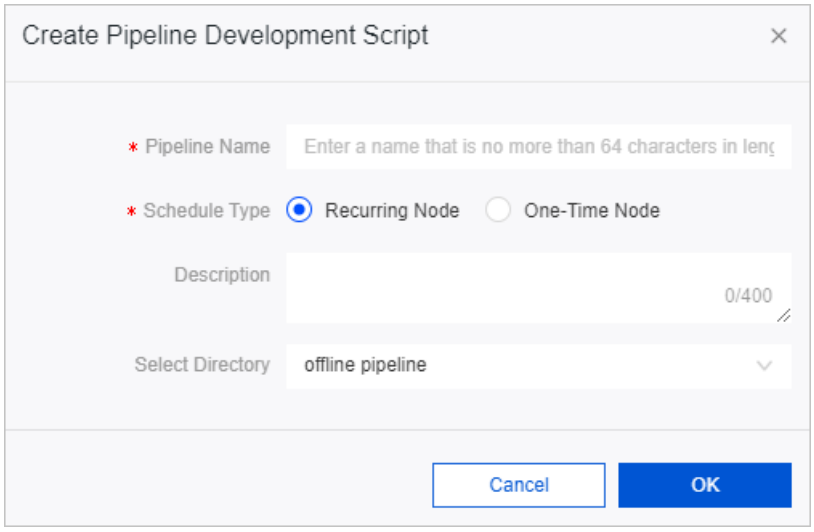
Context

An ADB for PostgreSQL input component is used to import data from an AnalyticDB for PostgreSQL database to Dataphin for integration and reprocessing.

Configure an ADB for PostgreSQL input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag an **ADB for PostgreSQL** component to the pipeline canvas on the left.
6. Right-click the **ADB for PostgreSQL** component and select **Configure Attributes**.
7. In the **ADB for PostgreSQL Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the input component. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an AnalyticDB for PostgreSQL data source.
Source Table	The quantity of source tables. Valid values: **Single Table** and **Multiple Tables**. **Note** If you select **Multiple Tables**, you must specify multiple source tables that have the same schema.
Table	The one or more source tables. Specify one or more source tables based on the **Source Table** parameter. ○ If you set the **Source Table** parameter to **Single Table**, click the icon and select a source table from the **Table** drop-down list. ○ If you set **Source Table** to **Multiple Tables**, perform the following steps to specify the source tables: a. Enter an expression in the Table field. Dataphin allows you to enter an enumeration expression, a regular-like expression, or a combination of both, for example, `table_[001-100];table_102`. b. Click the icon. c. In the **Confirm Match Details** dialog box, select the tables that match the expression you entered. d. Click **OK**.
Split Key	The shard key of the source table. You can specify a column in the source table as the shard key. We recommend that you use the primary key or an indexed column as the shard key. If you specify multiple source tables, you need to specify only one shard key because the source tables have the same schema.

Parameter	Description
Input Filtering	The filter condition for input fields. For example, you can enter `ds=${bizdate}`. The **Input Filtering** parameter is applicable to the following scenarios: ○ Specify a fixed portion of data. ○ Use parameters to filter data.
Output Fields	The output fields to be generated based on the input configuration. You can also perform the following operations to manage fields: ○ Click the icon in the **Actions** column to delete a field. ○ Click **Manage Fields**. In the **Manage Fields** dialog box, perform the following operations: ■ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ■ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section.

8. Click **OK**.

Copy an ADB for PostgreSQL input component

1. Right-click an ADB for PostgreSQL input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ADB for PostgreSQL input component

1. Right-click an ADB for PostgreSQL input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for an ADB for PostgreSQL input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click an ADB for PostgreSQL input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.6.16. Manage LogHub input components

This topic describes how to configure, copy, and delete a LogHub input component, and how to select a data sending mode for a LogHub input component.

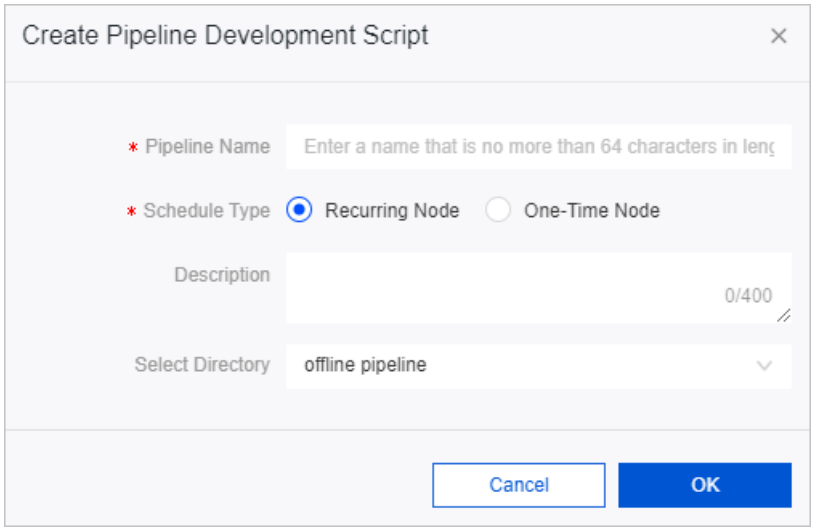
Context

A LogHub input component is used to import data from a LogHub data source to Dataphin for integration and reprocessing.

Configure a LogHub input component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Input**.
5. Drag a LogHub component to the pipeline canvas on the left.
6. Right-click the LogHub component and select **Configure Attributes**.
7. In the **Loghub Input Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the LogHub type. ○ The account that you use to configure the component has the read-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Logstore	The Logstore where the source data resides. All data in the Logstore is from the same data source.
Log StartTime	The start time of the period in which the required logs were generated. The value is in the format `yyyyMMddHHmmss`.
Log EndTime	The end time of the period in which the required logs were generated. The value is in the format `yyyyMMddHHmmss`.
Batch Number	The number of data records to read at a time. You can enter a number in the **Batch Number** field. Alternatively, you can click the upwards or downwards arrow in the field to adjust the value. Default value: 256. The system can read up to 1,000 data records at a time.

Parameter	Description
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format and click **OK**. Example: `[{ "index": 0, "name": "user_id", "type": "String" }, { "index": 1, "name": "user_name", "type": "String" }]` In the example, index, name, and type specify the sequence number, name, and data type of the output field, respectively. ○ Click **Create Output Field**. Set the **Field**, **Source Serial Number**, and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a LogHub input component

1. Right-click a LogHub input component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a LogHub input component

1. Right-click a LogHub input component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a LogHub input component

If an input component is connected to multiple components in the downstream, you must select a mode in which the data of the input component is sent to the downstream nodes.

1. Right-click a LogHub input component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.

- **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.7. Component library: Conversion components

Conversion components are used to convert the input data from an upstream component and transmit the converted data to downstream components. This topic describes how to configure the properties of the Field Selection, Field Computing, and Filtering components.

Go to the script development tab of a single offline pipeline. Click **Component Library** in the upper-right corner and then the icon before **Conversion** to open the **Conversion** list.

Field selection

A **Field Selection** component is used to filter the input fields from an upstream component, rename the fields, and change the order of output fields.

1. Drag and drop a **Field Selection** component to the pipeline canvas on the left.
2. Right-click the component box and select **Configure Attributes**.
3. In the **Field Selection Configuration** dialog box that appears, set relevant parameters as prompted and click **OK**.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** to perform the following operations in the dialog box that appears: ○ Select one or more fields and click the icon to move the selected fields from the **Selected Input Fields** list to the **Unselected Input Fields** list. ○ Select one or more fields and click the icon to move the selected fields from the **Unselected Input Fields** list to the **Selected Input Fields** list. You can also perform the following operations on existing fields: ○ Click the icon in the Actions column to edit an existing field. ○ Click the icon in the Actions column to delete an existing field.

Field computing

The applicable scenarios of a **Field Computing** component include but are not limited to:

- Split an input field from an upstream component into multiple fields.

- Generate derived fields based on the input fields from an upstream component.
- Map or de-identify the values of input fields from an upstream component.

To configure the properties of a Field Computing component, follow these steps:

1. Drag and drop a **Field Computing** component to the pipeline canvas on the left.
2. Right-click the component box and select **Configure Attributes**.
3. In the **Field Computing Configuration** dialog box that appears, set relevant parameters as prompted and click **OK**.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can create an output field by following these steps: i. Click **+ Create Field**. ii. In the **Create Field** dialog box that appears, set **Name** and **Expression** as prompted, and select a data type from the **Data Type** drop-down list. iii. Click **OK**. You can also click the icon in the Actions column to delete an existing field.

Filtering

A **Filtering** component is used to filter the input data from an upstream component and transmit the data that meets the specified filter conditions to output components.

1. Drag and drop a **Filtering** component to the pipeline canvas on the left.
2. Right-click the component box and select **Configure Attributes**.
3. In the **Conversion-Filtering Configuration** dialog box that appears, set relevant parameters as prompted and click **OK**.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.
Filter Conditions	The filter conditions for input fields. The following operators are supported: **=, !=, >, >=, <, <=, like, in, is null**, and **is not null**. For example, you can enter `user_id is not null and user_name like '%a%'` .
Output Fields	The output fields to be generated.

9.8.2.7.1. Manage Field Selection components

This topic describes how to configure, copy, and delete a Field Selection component, and how to select a data sending mode for a Field Selection component.

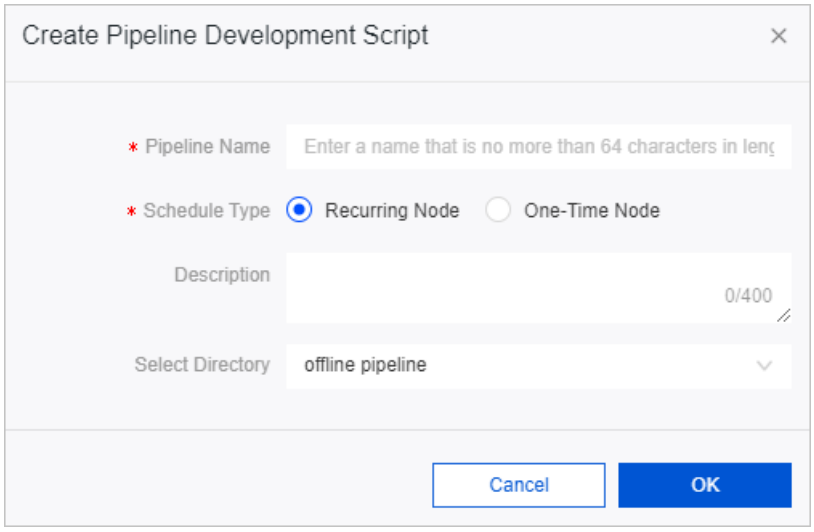
Context

A Field Selection component is used to filter the input fields from an upstream component, rename the fields, and change the order of output fields.

Configure a Field Selection component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Conversion**.
5. Drag a Field Selection component to the pipeline canvas on the left.
6. Right-click the Field Selection component and select **Configure Attributes**.
7. In the **Field Selection Configuration** dialog box, set the parameter as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.

Parameter	Description
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a Field Selection component

1. Right-click a Field Selection component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Field Selection component

1. Right-click a Field Selection component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a Field Selection component

If a conversion component is connected to multiple components in the downstream, you must select a mode in which the data of the conversion component is sent to the downstream nodes.

1. Right-click a Field Selection component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.7.2. Manage Field Computing components

This topic describes how to configure, copy, and delete a Field Computing component, and how to select a data sending mode for a Field Computing component.

Context

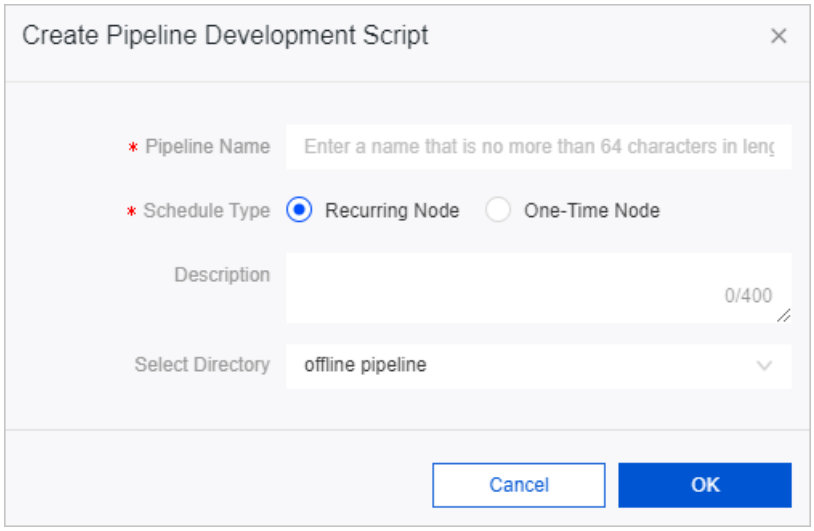
The applicable scenarios of a Field Computing component include but are not limited to:

- Split an input field from an upstream component into multiple fields.

- Generate derived fields based on the input fields from an upstream component.
- Map or de-identify the values of the input fields from an upstream component.

Configure a Field Computing component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.
   v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

   vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Conversion**.
5. Drag a Field Computing component to the pipeline canvas on the left.
6. Right-click the Field Computing component and select **Configure Attributes**.
7. In the **Field Computing Configuration** dialog box, set the parameter as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. To create an output field, perform the following steps: i. Click **Create Field**. ii. In the **Create Field** dialog box, set the **Name**, **Expression**, and **Data Type** parameters. When you set the **Expression** parameter, you can view the functions in the system and their usage in the **Function** section on the right. iii. Click **OK**. To delete an output field, click the icon in the **Actions** column.

8. Click **OK**.

Copy a Field Computing component

1. Right-click a Field Computing component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Field Computing component

1. Right-click a Field Computing component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a Field Computing component

If a conversion component is connected to multiple components in the downstream, you must select a mode in which the data of the conversion component is sent to the downstream nodes.

1. Right-click a Field Computing component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.7.3. Manage Filtering components

This topic describes how to configure, copy, and delete a Filtering component, and how to select a data sending mode for a Filtering component.

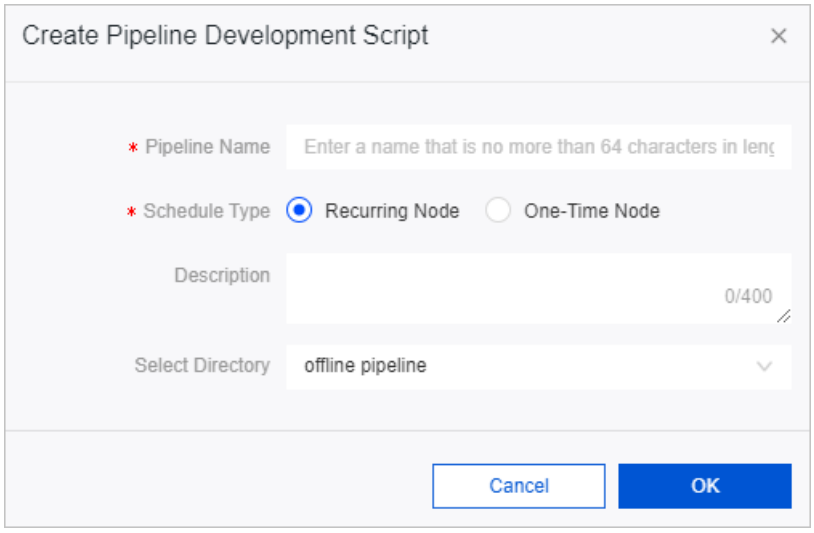
Context

A Filtering component is used to filter the input data from an upstream component and send the data that meets the specified filter conditions to output components.

Configure a Filtering component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Conversion**.
5. Drag a **Filtering** component to the pipeline canvas on the left.
6. Right-click the **Filtering** component and select **Configure Attributes**.
7. In the **Conversion-Filtering Configuration** dialog box, set the parameter as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.

Parameter	Description
Filter Conditions	The filter conditions for the input data. You can specify the filter conditions by using one of the following methods: ○ Perform the following steps: a. Click the icon in the **Field** column and select a field from the drop-down list. b. Click the icon in the **Operator** column and select an operator from the drop-down list. The following operators are supported: =, != , >, >=, <, <=, **LINKE**, **IS NULL**, and **IS NOT NULL**. c. Click the icon in the **Content** column and select **Table Field** or **Custom Recipients** from the drop-down list. ■ If you select **Table Field**, another drop-down list appears in the Content column. Select a field from the drop-down list. ■ If you select **Custom Recipients**, a field appears in the Content column. Enter a recipient in the field. Click the icon and then the icon. Select a data type from the drop-down list. d. Click the icon in the **Actions** column. The icon appears in a new line. Click this icon and select **AND** or **OR** from the drop-down list. ○ Click **Switch to Script Mode** and enter filter conditions in the code editor that appears. For example, you can enter `user_id is not null and user_name like '%a%'` . The following operators are supported: =, != , >, >=, <, <=, **LIKE**, **IS NULL**, and **IS NOT NULL**.
Output Fields	The output fields to be generated.

8. Click **OK**.

Copy a Filtering component

1. Right-click a Filtering component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Filtering component

1. Right-click a Filtering component and select **Delete**.
2. In the message that appears, click **OK**.

Select a data sending mode for a Filtering component

If a conversion component is connected to multiple components in the downstream, you must select a mode in which the data of the conversion component is sent to the downstream nodes.

1. Right-click a Filtering component and select **Data Sending Mode**.
2. Select a data sending mode.Two data sending modes are supported:
   - **Copy**: copies all the data of the upstream node to each downstream node.
   - **Distribute**: divides the data of the upstream node into equal shares based on the number of downstream nodes and distributes the data to the downstream nodes in turn. The total data volume of all the downstream nodes is equal to the data volume of the upstream node.

9.8.2.8. Component library: Process components

Process components are used in the data transmission process for throttling and conditional distribution. This topic describes how to configure the properties for throttling and conditional distribution.

Go to the script development tab of a single offline pipeline and open the component library.

Click the > icon before **Process** to open the **Process** list.

Throttling

An **Access Limit** component is used to control read/write operations on downstream service databases. To configure the properties for throttling, follow these steps:

1. Drag and drop an **Access Limit** component to the pipeline canvas on the left.
2. Right-click the component box and select **Configure Attributes**.
3. In the **Access Limit Configuration** dialog box that appears, set relevant parameters as prompted and click **OK**.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Speed Limit	The upper limit of the data transmission speed. ○ If you select **Data Volume-based Limit**, you can select a maximum data transmission rate. Valid values: **1M/s**, **2M/s**, **5M/s**, and **10M/s**. ○ If you select **Data Record-based Limit**, you must enter the maximum number of data records, for example, 30 records per second.

Conditional distribution

A **Conditional Distribution** component is used to split the input data from an upstream component and distribute the data to different downstream components.

1. Drag and drop a **Conditional Distribution** component to the pipeline canvas on the left.
2. Right-click the component box and select **Configure Attributes**.
3. In the **Conditional Distribution Configuration** dialog box that appears, set relevant parameters as prompted and click **OK**.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.
Merge Logic Configuration	The conditions for distribution. The following operators are supported: =, !=, >, >=, <, <=, like, in, is null, and is not null. For example, you can enter `user_id is not null and user_name like '%a%'`. Currently, distribution is only based on the results of true and false. If you do not specify the conditions for distribution, the result defaults to true.
Output Fields	The output fields to be generated. You can click **Manage Fields** to perform the following operations in the dialog box that appears: ○ Select one or more fields and click the icon to move the selected fields from the **Selected Input Fields** list to the **Unselected Input Fields** list. ○ Select one or more fields and click the icon to move the selected fields from the **Unselected Input Fields** list to the **Selected Input Fields** list. You can also perform the following operations on existing fields: ○ Click the icon in the Actions column to edit an existing field. ○ Click the icon in the Actions column to delete an existing field.

9.8.2.8.1. Manage Access Limit components

This topic describes how to configure, copy, and delete an Access Limit component.

Context

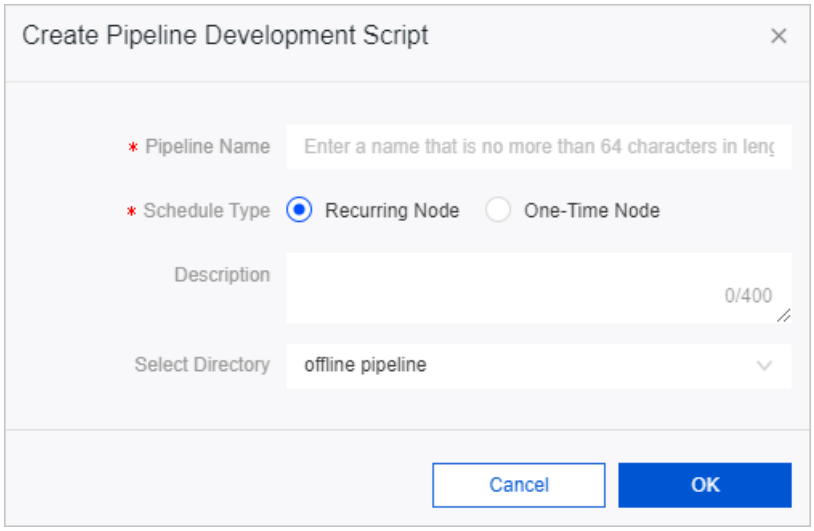
An Access Limit component is used to control read/write operations on downstream databases.

Configure an Access Limit component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.

iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Process**.
5. Drag an Access Limit component to the pipeline canvas on the left.
6. Right-click the Access Limit component and select **Configure Attributes**.
7. In the **Access Limit Configuration** dialog box, set the parameter as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Speed Limit	The upper limit of the data transmission speed. ○ If you select **Data Volume-based Limit**, you can select a maximum data transmission rate. Valid values: **1M/s**, **2M/s**, **5M/s**, and **10M/s**. ○ If you select **Data Record-based Limit**, you must enter the maximum number of data records that can be sent per second, for example, 30 records per second.

8. Click **OK**.

Copy an Access Limit component

1. Right-click an Access Limit component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an Access Limit component

1. Right-click an Access Limit component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.8.2. Manage Conditional Distribution components

This topic describes how to configure, copy, and delete a Conditional Distribution component.

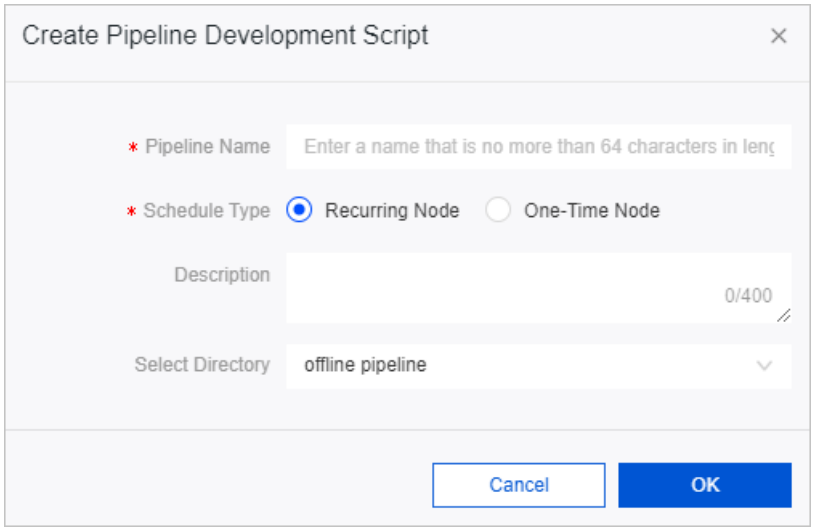
Context

A Conditional Distribution component is used to split the input data from an upstream component and distribute the data to different downstream components.

Configure a Conditional Distribution component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Process**.
5. Drag a Conditional Distribution component to the pipeline canvas on the left.
6. Right-click the Conditional Distribution component and select **Configure Attributes**.
7. In the **Conditional Distribution Configuration** dialog box, set the parameter as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Input Fields	The input fields based on the upstream input.

Parameter	Description
Merge Logic Configuration	The conditions for distribution. The following operators are supported: =, != , >, >=, <, <=, LIKE, IN, IS NULL, and IS NOT NULL. For example, you can enter `user_id is not null and user_name like '%a%'` . Distribution is only based on the results of true and false. If you do not specify the conditions for distribution, the result defaults to true.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section. ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

8. Click **OK**.

Copy a Conditional Distribution component

1. Right-click a Conditional Distribution component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Conditional Distribution component

1. Right-click a Conditional Distribution component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9. Output components

9.8.2.9.1. Manage MySQL output components

This topic describes how to configure, copy, and delete a MySQL output component.

Context

A MySQL output component is used to write data processed by Dataphin to a MySQL database for data consumption.

Configure a MySQL output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

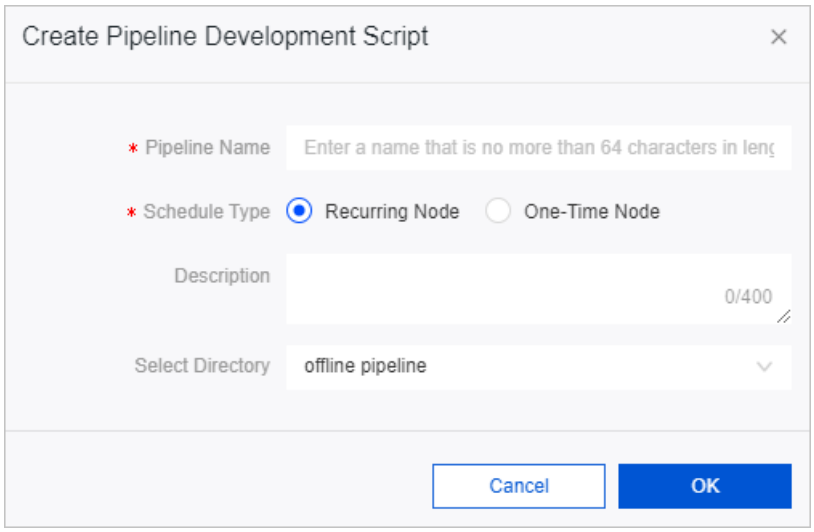
i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.

iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Output**.
5. Drag a MySQL component to the pipeline canvas on the left.
6. Right-click the MySQL component and select **Configure Attributes**.
7. In the **MySQL Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source.
Table	The destination table.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Parse Solution	Optional. The operations before and after data output. For more information, move the pointer over **Component Description** in the upper-right corner of the dialog box.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a MySQL output component

1. Right-click a MySQL output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a MySQL output component

1. Right-click a MySQL output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.2. Manage DRDS output components

This topic describes how to configure, copy, and delete a DRDS output component.

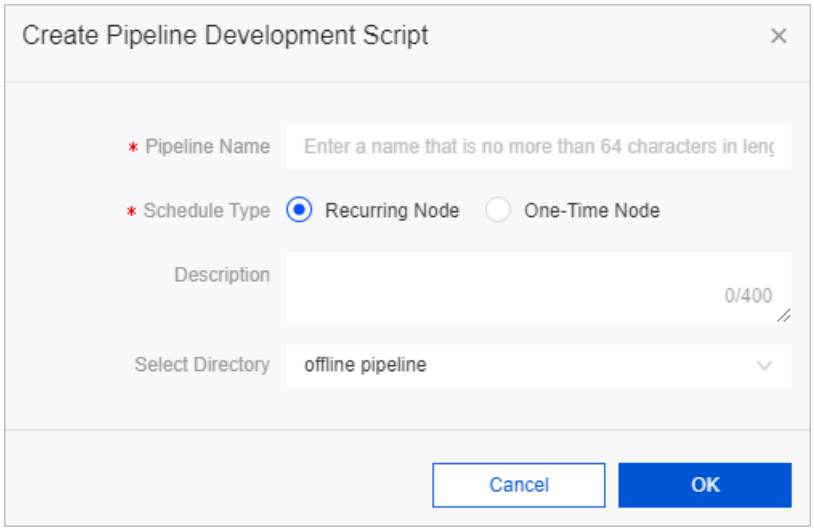
Context

A DRDS output component is used to write data processed by Dataphin to a PolarDB-X database for data consumption.

Configure a DRDS output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag a DRDS component to the pipeline canvas on the left.
6. Right-click the DRDS component and select **Configure Attributes**.
7. In the **DRDS Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to Data Source to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a DRDS output component

1. Right-click a DRDS output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a DRDS output component

1. Right-click a DRDS output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.3. Manage Maxcompute output components

This topic describes how to configure, copy, and delete a Maxcompute output component.

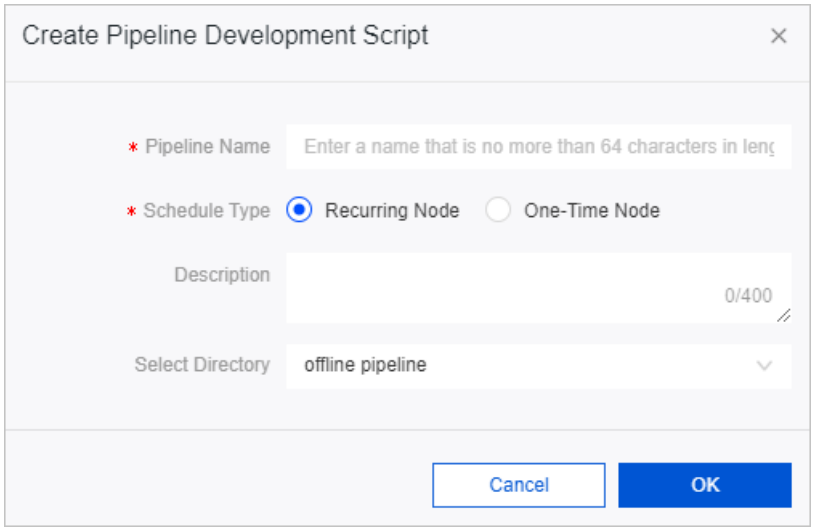
Context

A Maxcompute output component is used to write data processed by Dataphin to a MaxCompute project for data consumption.

Configure a Maxcompute output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Output**.
5. Drag a Maxcompute component to the pipeline canvas on the left.
6. Right-click the Maxcompute component and select **Configure Attributes**.
7. In the **Maxcompute Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table. You can also perform the following steps to create a destination table: i. Click **Generate Target Table by One Click**. ii. In the dialog box that appears, write an SQL script as required and click **Create**. By default, the table created is the destination table for the output data.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Partitioning	The partition information required if the destination table is a partitioned table. ○ If the table you selected is a partitioned table, you must enter partition information, for example, ds=${bizdate}. ○ If the table you selected is not a partitioned table, you do not need to set this parameter.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.

Parameter	Description
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a Maxcompute output component

1. Right-click a Maxcompute output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Maxcompute output component

1. Right-click a Maxcompute output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.4. Manage SQL Server output components

This topic describes how to configure, copy, and delete an SQL Server output component.

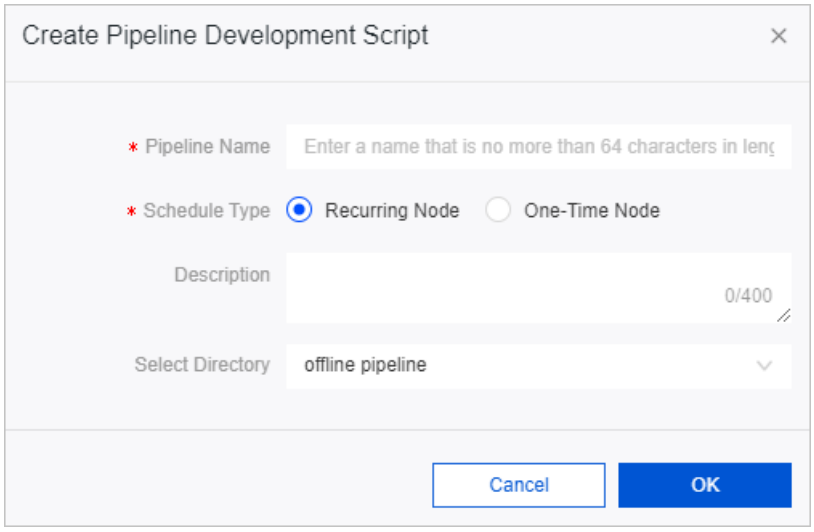
Context

An SQL Server output component is used to write data processed by Dataphin to an SQL Server database for data consumption.

Configure an SQL Server output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an SQL Server component to the pipeline canvas on the left.
6. Right-click the SQL Server component and select **Configure Attributes**.
7. In the **SQL Server Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements. ○ **Enter Preparation Statement**: the SQL statements to execute before data output. ○ **Enter Completion Statement**: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**. ○ **Position-based Mapping**: The fields in the **Input Fields** section and those in the **Output Fields** section are mapped based on their positions. ○ **Name-based Mapping**: The fields in the **Input Fields** section and those in the **Output Fields** section are mapped based on their names.

8. Click **OK**.

Copy an SQL Server output component

1. Right-click an SQL Server output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an SQL Server output component

1. Right-click an SQL Server output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.5. Manage ORACLE output components

This topic describes how to configure, copy, and delete an ORACLE output component.

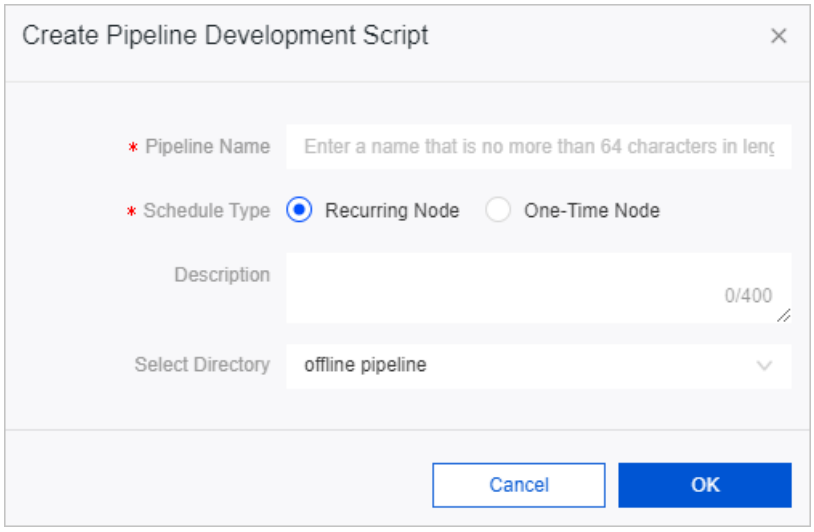
Context

An ORACLE output component is used to write data processed by Dataphin to an Oracle database for data consumption.

Configure an ORACLE output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an ORACLE component to the pipeline canvas on the left.
6. Right-click the ORACLE component and select **Configure Attributes**.
7. In the **Oracle Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the properties has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements. ○ **Enter Preparation Statement**: the SQL statements to execute before data output. ○ **Enter Completion Statement**: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an ORACLE output component

1. Right-click an ORACLE output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ORACLE output component

1. Right-click an ORACLE output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.6. Manage PostgreSQL output components

This topic describes how to configure, copy, and delete a PostgreSQL output component.

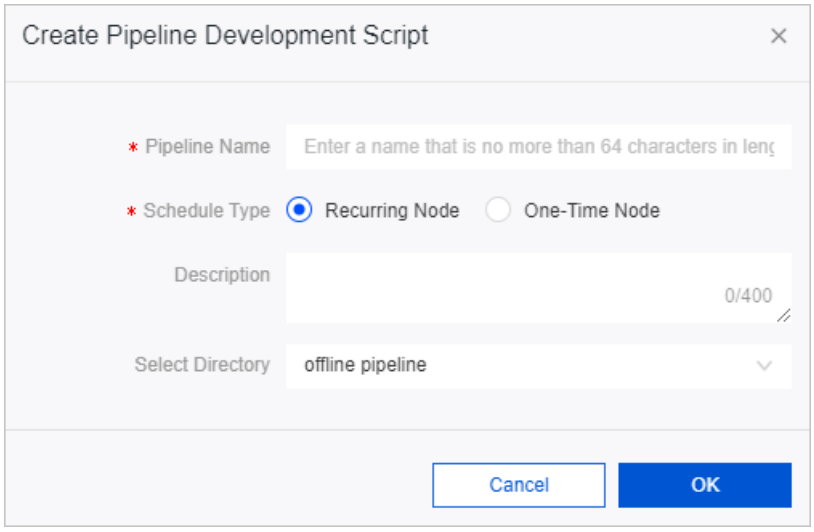
Context

A PostgreSQL output component is used to write data processed by Dataphin to a PostgreSQL database for data consumption.

Configure a PostgreSQL output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag a PostgreSQL component to the pipeline canvas on the left.
6. Right-click the PostgreSQL component and select **Configure Attributes**.
7. In the **PostgreSQL Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to Data Source to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click Enter Preparation Statement or Enter Completion Statement next to Parse Solution and enter SQL statements. ○ Enter Preparation Statement: the SQL statements to execute before data output. ○ Enter Completion Statement: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click Manage Fields and perform the following operations in the dialog box that appears: ○ Select one or more fields in the Selected Input Fields section and click the icon to move the fields to the Unselected Input Fields section. ○ Select one or more fields in the Unselected Input Fields section and click the icon to move the fields to the Selected Input Fields section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click Quick Mapping and select Position-based Mapping or Name-based Mapping.

8. Click **OK**.

Copy a PostgreSQL output component

1. Right-click a PostgreSQL output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a PostgreSQL output component

1. Right-click a PostgreSQL output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.7. Manage Vertica output components

This topic describes how to configure, copy, and delete a Vertica output component.

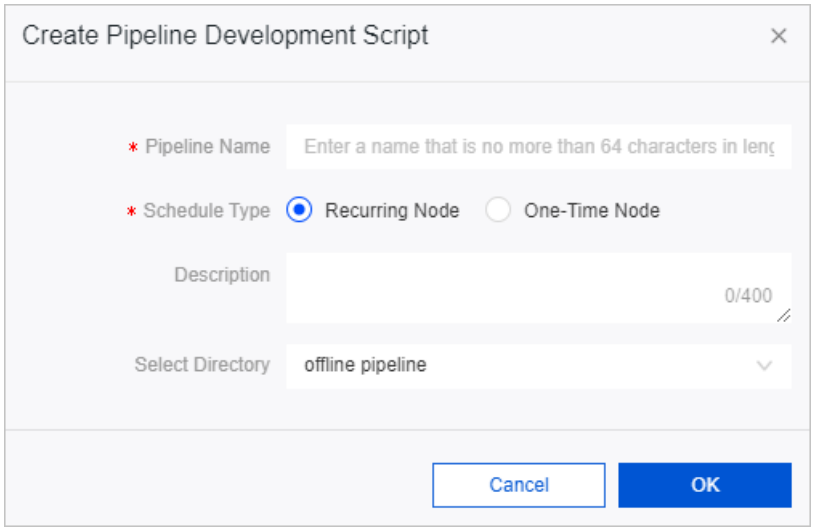
Context

A Vertica output component is used to write data processed by Dataphin to a Vertica database for data consumption.

Configure a Vertica output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the icon before **Output**.
5. Drag a Vertica component to the pipeline canvas on the left.
6. Right-click the Vertica component and select **Configure Attributes**.
7. In the **Vertica Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements. ○ **Enter Preparation Statement**: the SQL statements to execute before data output. ○ **Enter Completion Statement**: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a Vertica output component

1. Right-click a Vertica output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Vertica output component

1. Right-click a Vertica output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.8. Manage FTP output components

This topic describes how to configure, copy, and delete an FTP output component.

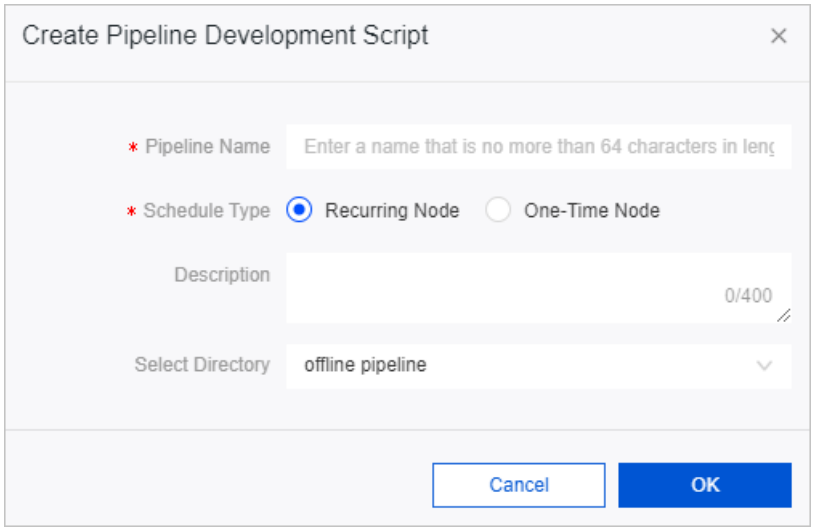
Context

An FTP output component is used to write data processed by Dataphin to an FTP server for data consumption.

Configure an FTP output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an FTP component to the pipeline canvas on the left.
6. Right-click the FTP component and select **Configure Attributes**.
7. In the **FTP Output Configuration** dialog box, set the parameters as required.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the  icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
File Path	The path of the destination file.
File Type	The type of the destination file. Valid values: **TexT** and **CSV**.
File Encoding	The encoding format of the destination file. Valid values: **UTF-8** and **GBK**.
Field Delimiter	Optional. The delimiter that is used to separate fields in the destination file. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Loading Policy	The policy for writing data to the destination file. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination file. ○ **Append Data**: appends the source data to the destination file, without modifying the existing data in the destination file.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "name":"cf1:a", "type":"String" },` `{ "name":"cf1:b", "type":"String" }]` ○ Click **Create Output Field**. Set the **Field** and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the  icon in the **Actions** column to edit a field. ○ Click the  icon in the **Actions** column to delete a field.

Parameter	Description
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an FTP output component

1. Right-click an FTP output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an FTP output component

1. Right-click an FTP output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.9. Manage HDFS output components

This topic describes how to configure, copy, and delete an HDFS output component.

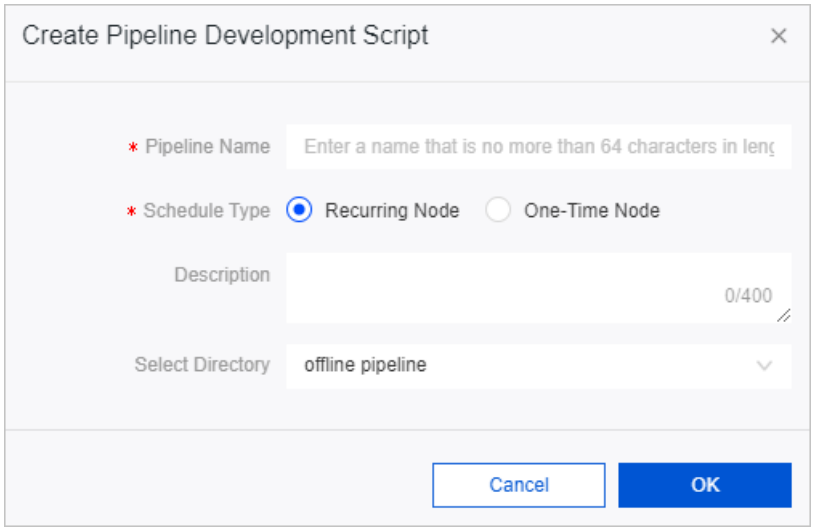
Context

An HDFS output component is used to write data processed by Dataphin to an HDFS for data consumption.

Configure an HDFS output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an HDFS component to the pipeline canvas on the left.
6. Right-click the HDFS component and select **Configure Attributes**.
7. In the **HDFS Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
File Path	The path of the destination file.
File Type	The type of the destination file. Valid values: **TexT** and **ORC**.
File Encoding	The encoding format of the destination file. Valid values: **UTF-8** and **GBK**.
Field Delimiter	Optional. The delimiter that is used to separate fields in the destination file. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Loading Policy	The policy for writing data to the destination file. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination file. ○ **Append Data**: appends the source data to the destination file, without modifying the existing data in the destination file.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "name":"cf1:a", "type":"String" },` `{ "name":"cf1:b", "type":"String" }]` ○ Click **Create Output Field**. Set the **Field** and **Type** parameters as prompted. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.

Parameter	Description
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an HDFS output component

1. Right-click an HDFS output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an HDFS output component

1. Right-click an HDFS output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.10. Manage MongoDB output components

This topic describes how to configure, copy, and delete a MongoDB output component.

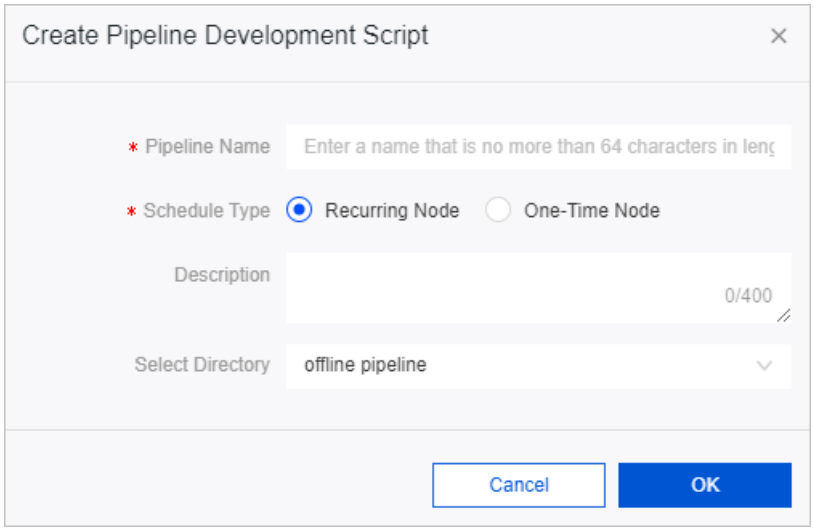
Context

A MongoDB output component is used to write data processed by Dataphin to a MongoDB database for data consumption.

Configure a MongoDB output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag a MongoDB component to the pipeline canvas on the left.
6. Right-click the MongoDB component and select **Configure Attributes**.
7. In the **MongoDB Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MongoDB data source.
Table	The destination table.
Update Information	The update information of fields. For example, you can enter `{"isUpsert":"true","upsertkey""unique_id"}` .
Field Delimiter	The delimiter that is used to separate fields in the destination table. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "name":"cf1:a", "type":"String" },` `{ "name":"cf1:b", "type":"String" }]` ○ Click **Create Output Field**. Set the **Field** and **Type** parameters as prompted. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a MongoDB output component

1. Right-click a MongoDB output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a MongoDB output component

1. Right-click a MongoDB output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.11. Manage Hbase output components

This topic describes how to configure, copy, and delete an Hbase output component.

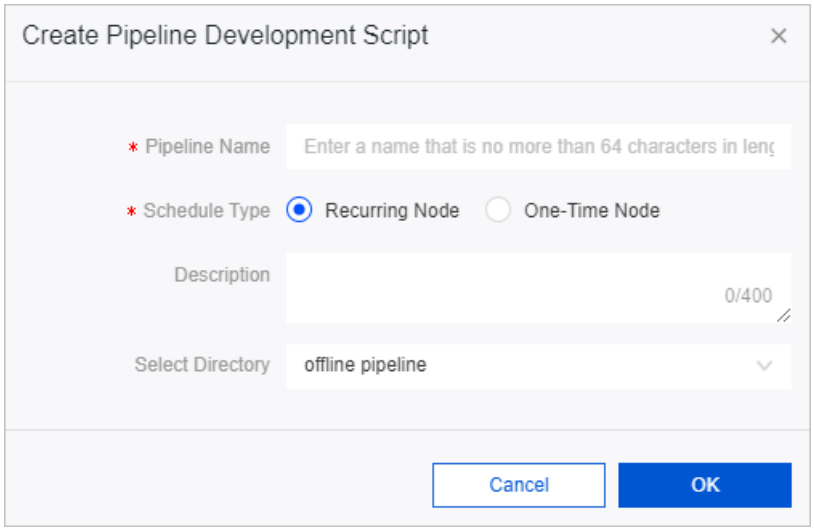
Context

An Hbase output component is used to write data processed by Dataphin to an HBase database for data consumption.

Configure an Hbase output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an Hbase component to the pipeline canvas on the left.
6. Right-click the Hbase component and select **Configure Attributes**.
7. In the **HBase Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the  icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
File Encoding	The encoding format of the destination table. Valid values: **UTF-8** and **GBK**.
Version Column	The timestamp of data written to the destination table in HBase. For example, you can enter {"index":-1,"value":123456789}.
Rowkey	The rowkey rule of the destination table. The rowkey is composed of several output fields that are connected by an underscore (_). Example: `[{"index":0,"type":"string"},{"index":2,"type":"string"},{"index":-1,"string","value":"_"}]` In this example, the rowkey is composed of the first and third output fields that are connected by an underscore (_).
Input Fields	The input fields based on the upstream input.

Parameter	Description
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "name": "col_integer", "type": "integer" }, { "name": "col_long", "type": "long" }, { "name": "col_double", "type": "double" }]` In the example, name and type specify the name and data type of the output field, respectively. ○ Click **Create Output Field**. Set the **Column Family**, **Field**, and **Type** parameters as prompted. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select Position-based Mapping or **Name-based Mapping**.

8. Click **OK**.

Copy an Hbase output component

1. Right-click an Hbase output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an Hbase output component

1. Right-click an Hbase output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.12. Manage Hive output components

This topic describes how to configure, copy, and delete a Hive output component.

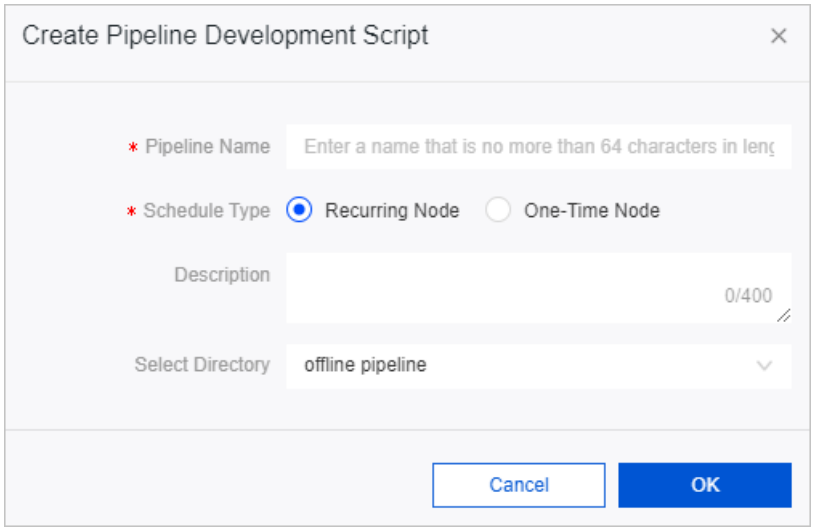
Context

A Hive output component is used to write data processed by Dataphin to a Hive database for data consumption.

Configure a Hive component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag a Hive component to the pipeline canvas on the left.
6. Right-click the Hive component and select **Configure Attributes**.
7. In the **Hive Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the  icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a Hive data source.
Table	The destination table.
File Encoding	The encoding format of the destination table. Valid values: **UTF-8** and **GBK**.
Field Delimiter	Optional. The delimiter that is used to separate fields in the destination table. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Compression Format	The compression format of the destination table. Valid values: ○ **zip** ○ **gzip** ○ **bzip2**
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the  icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the  icon to move the fields to the **Selected Input Fields** section.

Parameter	Description
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy a Hive component

1. Right-click a Hive component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete a Hive component

1. Right-click a Hive component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.13. Manage ElasticSearch output components

This topic describes how to configure, copy, and delete an ElasticSearch output component.

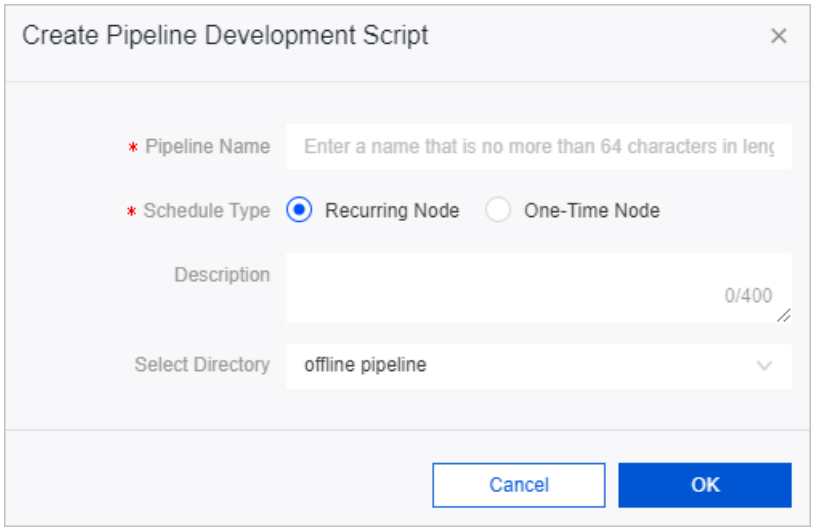
Context

An ElasticSearch output component is used to write data processed by Dataphin to an Elasticsearch database for data consumption.

Configure an ElasticSearch output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an ElasticSearch component to the pipeline canvas on the left.
6. Right-click the ElasticSearch component and select **Configure Attributes**.
7. In the **ElasticSearch Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Index	The index of the destination table.
Index Type	The index type of the destination table.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Field Delimiter	Optional. The delimiter that is used to separate fields in the destination table. If you do not specify this parameter, a comma (,) is used as the field delimiter.
Input Fields	The input fields based on the upstream input.

Parameter	Description
Output Fields	The output fields to be generated. You can create output fields by using one of the following methods: ○ Click **Batch Create**. In the Batch Create in JSON Format dialog box, configure multiple output fields in the JSON format. Example: `[{ "name": "col_integer", "type": "integer" }, { "name": "col_long", "type": "long" }, { "name": "col_double", "type": "double" }]` In the example, name and type specify the name and data type of the output field, respectively. ○ Click **Create Output Field**. Set the **Field** and **Type** parameters as required. You can also perform the following operations on existing fields: ○ Click the icon in the **Actions** column to edit a field. ○ Click the icon in the **Actions** column to delete a field.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an ElasticSearch output component

1. Right-click an ElasticSearch output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ElasticSearch output component

1. Right-click an ElasticSearch output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.14. Manage ADB for MySQL2.0 output components

This topic describes how to configure, copy, and delete an ADB for MySQL2.0 output component.

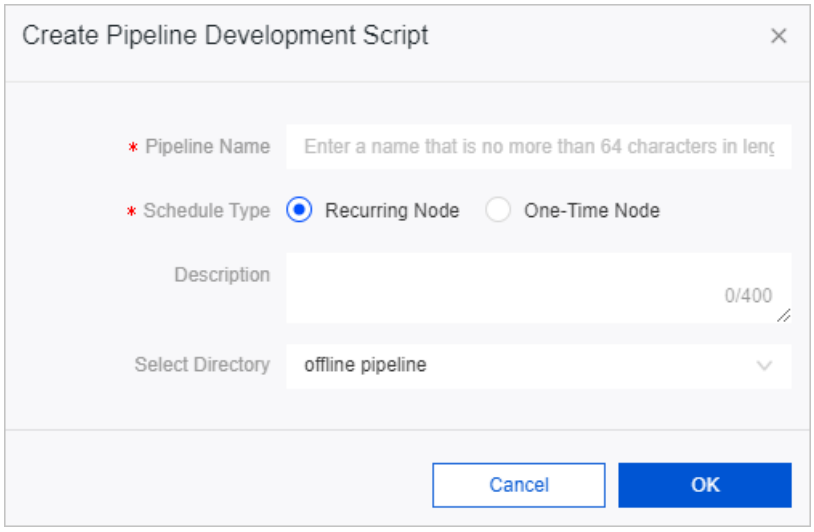
Context

An ADB for MySQL2.0 output component is used to write data processed by Dataphin to an AnalyticDB for MySQL 2.0 database for data consumption.

Configure an ADB for MySQL2.0 output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an **ADB for MySQL2.0** component to the pipeline canvas on the left.
6. Right-click the **ADB for MySQL2.0** component and select **Configure Attributes**.
7. In the **AnalyticDB for MySQL 2.0 Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the  icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create an AnalyticDB for MySQL V3.0 data source.
Table	The destination table.
Mode	The mode for writing data to the destination table. Valid values: ○ **Insert Mode**: applicable when fewer than 10 million data records are to be written. If you select this mode, perform the following operations to set the **Parse Solution** parameter: ■ Click **Enter Preparation Statement** and enter the SQL statements to execute before data output. ■ Click **Enter Completion Statement** and enter the SQL statements to execute after data output. ○ **Load Mode**: applicable when more than 10 million data records are to be written. If you select this mode, you must set the following parameters: ■ **Load policy**: the policy for writing data to the destination table. Valid values: **Append Data** and **Overwrite Data**. ■ **Load Parameters**: the parameters for connecting to MaxCompute. Example: `{"accessid":"<yourAccessKeyId>","accessKey":"*<yourAccessKeySecret>","odpsServer":"****", "tunnelServer":"****","accountType":"aliyun","project":"transfer_project"}` ■ **Alibaba Cloud Account**: the Alibaba Cloud account that is authorized to write data, for example, ALIYUN$garuda_data@aliyun.com.
Input Fields	The input fields based on the upstream input.

Parameter	Description
Output Fields	The output fields to be generated. You can click **Manage Fields** and perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an ADB for MySQL2.0 output component

1. Right-click an ADB for MySQL2.0 output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ADB for MySQL2.0 output component

1. Right-click an ADB for MySQL2.0 output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.15. Manage ADB for MySQL3.0 output components

This topic describes how to configure, copy, and delete an ADB for MySQL3.0 output component.

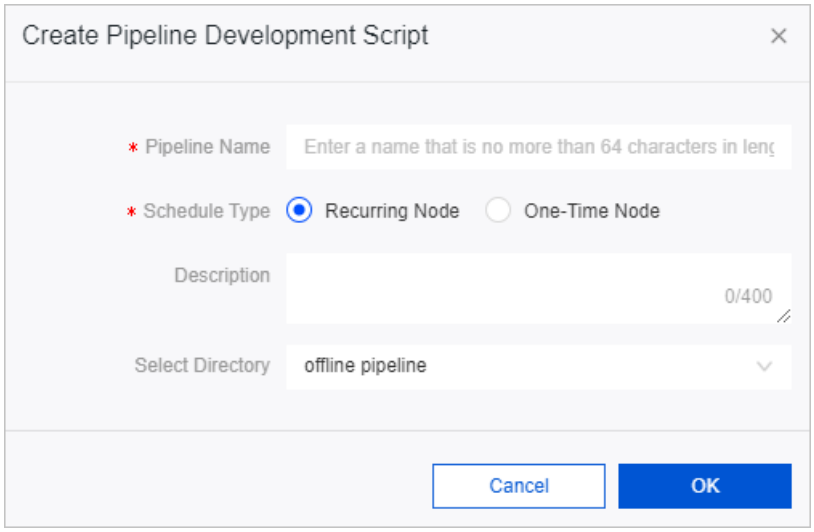
Context

An ADB for MySQL3.0 output component is used to write data processed by Dataphin to an AnalyticDB for MySQL 3.0 database for data consumption.

Configure an ADB for MySQL3.0 output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an ADB for MySQL3.0 component to the pipeline canvas on the left.
6. Right-click the ADB for MySQL3.0 component and select **Configure Attributes**.
7. In the **AnalyticDB for MySQL 3.0 Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source.
Table	The destination table.
Loading Policy	The policy for writing data to the destination table. Valid values: ○ **Overwrite Data**: uses the source data to overwrite the existing data in the destination table. ○ **Append Data**: appends the source data to the destination table, without modifying the existing data in the destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements. ○ **Enter Preparation Statement**: the SQL statements to execute before data output. ○ **Enter Completion Statement**: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** to perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an ADB for MySQL3.0 output component

1. Right-click an ADB for MySQL3.0 output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ADB for MySQL3.0 output component

1. Right-click an ADB for MySQL3.0 output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.2.9.16. Manage ADB for PostgreSQL output components

This topic describes how to configure, copy, and delete an ADB for PostgreSQL output component.

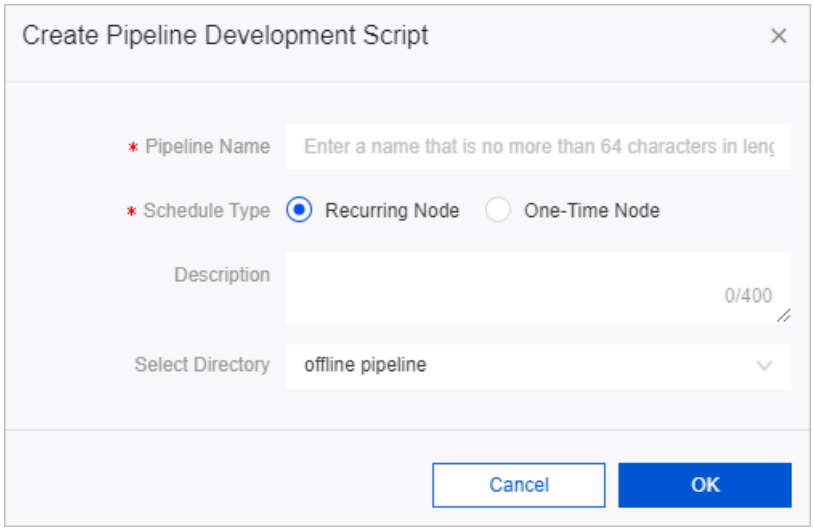
Context

An ADB for PostgreSQL output component is used to write data processed by Dataphin to an AnalyticDB for PostgreSQL database for data consumption.

Configure an ADB for PostgreSQL output component

1. Log on to the Dataphin console.
2. Perform the following steps to create an offline migration pipeline:
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. In the top navigation bar, move the pointer over **Develop** and select **Integrated**. The **Integrated** page appears.
   iv. On the **Integrated** page, move the pointer over the icon next to **Script** and select **Offline Single Pipeline**.

v. In the **Create Pipeline Development Script** dialog box, set the parameters as required.



Parameter	Description
Pipeline Name	The name of the offline migration pipeline to be created.
Schedule Type	The scheduling type of the task corresponding to the offline migration pipeline. Valid values: ■ **Recurring Node**: a task that is run on a specified schedule. ■ **One-Time Node**: a task that has no dependencies and is triggered after you click Run.
Description	The description of the offline migration pipeline.
Select Directory	The folder where the offline migration pipeline resides.

vi. Click **OK**.

3. On the configuration tab of the pipeline, click **Component Library** in the upper-right corner.
4. Click the > icon before **Output**.
5. Drag an **ADB for PostgreSQL** component to the pipeline canvas on the left.
6. Right-click the **ADB for PostgreSQL** component and select **Configure Attributes**.
7. In the **AnalyticDB for PostgreSQL Output Configuration** dialog box, set the parameters as prompted.

Parameter	Description
Step Name	The name of the component. Enter a name based on the scenario.

Parameter	Description
Data Source	The data source of the component. Select a data source that has been configured in Dataphin. The data source must meet the following requirements: ○ The data source is of the same type as the output component. ○ The account that you use to configure the component has the write-through permission on the data source. If your account does not have the required permission, apply for permissions on the data source. For more information, see Apply for permissions on a data source. You can also click the icon next to **Data Source** to go to the Data Sources page and create a data source. For more information, see Create a MySQL data source.
Table	The destination table.
Parse Solution	Optional. The operations before and after data output. To specify operations before or after data output, click **Enter Preparation Statement** or **Enter Completion Statement** next to **Parse Solution** and enter SQL statements. ○ **Enter Preparation Statement**: the SQL statements to execute before data output. ○ **Enter Completion Statement**: the SQL statements to execute after data output.
Input Fields	The input fields based on the upstream input.
Output Fields	The output fields to be generated. You can click **Manage Fields** to perform the following operations in the dialog box that appears: ○ Select one or more fields in the **Selected Input Fields** section and click the icon to move the fields to the **Unselected Input Fields** section. ○ Select one or more fields in the **Unselected Input Fields** section and click the icon to move the fields to the **Selected Input Fields** section.
Mapping	The mapping between the input and output fields. To specify mappings one by one, click a field in the input field list and a field in the output field list. To specify multiple mappings at a time, click **Quick Mapping** and select **Position-based Mapping** or **Name-based Mapping**.

8. Click **OK**.

Copy an ADB for PostgreSQL output component

1. Right-click an ADB for PostgreSQL output component and select **Copy**.
2. Right-click a blank area on the pipeline canvas and select **Paste**.

Delete an ADB for PostgreSQL output component

1. Right-click an ADB for PostgreSQL output component and select **Delete**.
2. In the message that appears, click **OK**.

9.8.3. Data ingestion

9.8.3.1. Create a folder for storing sync tasks

You can use different folders to manage different types of sync tasks. This topic describes how to create a folder for storing sync tasks and the operations that you can perform on created folders.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The Sync Tasks section appears.
6. In the **Sync Tasks** section, click the icon next to **Sync Tasks**.
7. In the **Create Folder** dialog box, set the **Name** and **Select Directory** parameters.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Sync Tasks section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Sync Tasks section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.

Operation	Description
Delete a folder	To delete a folder, perform the following steps: i. In the Sync Tasks section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.8.3.2. Create a destination table for data synchronization

A destination table for data synchronization is a table that Dataphin uses to process business data. It is in the computing engine bound to the project to which the business belongs. This topic describes how to create a destination table for data synchronization.

Context

Dataphin allows you to create a destination table for data synchronization by creating an ad hoc query task or a batch processing task.

Create a destination table by creating an ad hoc query task

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.
5. Create a folder for storing ad hoc query tasks.
   i. In the **Ad Hoc Query** section, click the icon next to **Ad Hoc Query**.
   ii. In the **Create Folder** dialog box, set the **Name** and Select Directory parameters.
   iii. Click **OK**.
6. Create an ad hoc query task.
   i. In the **Ad Hoc Query** section, click the icon next to **Ad Hoc Query**.
   ii. In the **Create Item** dialog box, set the **Name**, **Description**, and **Select Directory** parameters.
   iii. Click **OK**. The Code Editor tab appears.
7. Write the code of the ad hoc query task on the Code Editor tab.

i. Write SQL statements to create a destination table. In this example, write the following SQL statements:

```
-- Table<store_sales (23 cols)  partition=ss_sold_date_sk>
drop table if exists store_sales;
create table if not exists store_sales(
    ss_sold_date_sk bigint
,   ss_sold_time_sk bigint
,   ss_item_sk bigint
,   ss_customer_sk bigint
,   ss_cdemo_sk bigint
,   ss_hdemo_sk bigint
,   ss_addr_sk bigint
,   ss_store_sk bigint
,   ss_promo_sk bigint
,   ss_ticket_number bigint
,   ss_quantity int
,   ss_wholesale_cost double
,   ss_list_price double
,   ss_sales_price double
,   ss_ext_discount_amt double
,   ss_ext_sales_price double
,   ss_ext_wholesale_cost double
,   ss_ext_list_price double
,   ss_ext_tax double
,   ss_coupon_amt double
,   ss_net_paid double
,   ss_net_paid_inc_tax double
,   ss_net_profit double
)
partitioned by (ds string)
```

ii. Click **Precompile** in the upper-right corner of the Code Editor tab to check whether the SQL statement complies with the standard.If the SQL statements do not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statements so that they comply with the standard.

iii. Click **Run** in the upper-right corner of the Code Editor tab to execute the SQL statements. Then, check whether the SQL statements are executed based on the information on the Console tab in the lower part of the page.

8. If the SQL statements are executed, click the icon in the upper-right corner of the Code Editor tab.

Create a destination table by creating a batch processing task

1. On the Dataphin homepage, click **R&D** in the top navigation bar.
2. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
3. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
4. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The Batch Processing Tasks section appears.
5. Create a folder for storing batch processing tasks.
   i. In the **Batch Processing Tasks** section, click the icon next to **Batch Processing Tasks**.
   ii. In the **Create Folder** dialog box, set the **Name** and Select Directory parameters.
   iii. Click **OK**.
6. Create a batch processing task of the MAX_COMPUTE_SQL type.
   i. In the **Batch Processing Tasks** section, click the icon next to **Batch Processing Tasks** and select **MAX_COMPUTE_SQL**.
   ii. In the **Create Item** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the batch processing task, for example, SQL.
Schedule Type	The scheduling type of the task. Valid values: **Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   iii. Click **OK**. The Code Editor tab appears.
7. Write the code of the batch processing task on the Code Editor tab.

i. Write SQL statements to create a destination table. In this example, write the following SQL statements:

```
-- Table<customer (18 cols)>


drop table if exists customer;
create table if not exists customer(
    c_customer_sk bigint
,   c_customer_id string
,   c_current_cdemo_sk bigint
,   c_current_hdemo_sk bigint
,   c_current_addr_sk bigint
,   c_first_shipto_date_sk bigint
,   c_first_sales_date_sk bigint
,   c_salutation string
,   c_first_name string
,   c_last_name string
,   c_preferred_cust_flag string
,   c_birth_day int
,   c_birth_month int
,   c_birth_year int
,   c_birth_country string
,   c_login string
,   c_email_address string
,   c_last_review_date_sk bigint
)
;
```

ii. Click **Precompile** in the upper-right corner of the Code Editor tab to check whether the SQL statement complies with the standard.If the SQL statements do not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statements so that they comply with the standard.

iii. Click **Run** in the upper-right corner of the Code Editor tab to execute the SQL statements. Then, check whether the SQL statements are executed based on the information on the Console tab in the lower part of the page.

8. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
9. Save, submit, and publish the batch processing task.

i. Click the icon in the upper-right corner of the Code Editor tab to save the task.

ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.

iii. (Optional)Publish the batch processing task.

- If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.8.3.3. Create a sync task

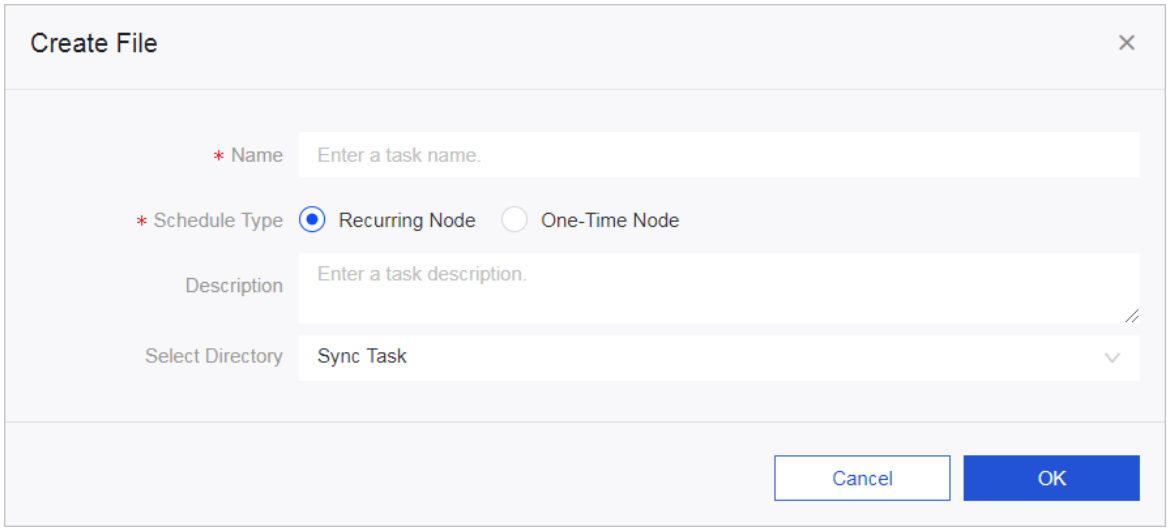
This topic describes how to create a sync task for synchronizing data from a business data source to a destination table in Dataphin.

Prerequisites

Destination tables for data synchronization are created. For more information, see Create a destination table for data synchronization.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The Sync Tasks section appears.
6. Create a folder for storing sync tasks.
   i. In the **Sync Tasks** section, click the icon next to **Sync Tasks**.
   ii. In the **Create Folder** dialog box, set the **Name** and Select Directory parameters.
   iii. Click **OK**.
7. In the **Sync Tasks** section, click the icon next to **Sync Tasks**.
8. In the **Create Item** dialog box, set the parameters as required.



Parameter	Description
Name	The name of the sync task.
Schedule Type	The scheduling type of the task. Valid values: ○ **Recurring Node**: If you select this option, you must configure the scheduling policy. ○ **One-Time Node**: If you select this option, you do not need to configure the scheduling policy.
Description	The description of the task.
Select Directory	The directory for storing the task.

9. Click **OK**. The sync task is created and is in the **Draft** state.

   The following table describes the operations that you can perform on sync tasks in the **Draft** state.

Operation	Description
Modify a sync task	To modify a sync task, perform the following steps: i. In the **Sync Tasks** section, move the pointer over the icon next to the sync task that you want to modify and select **Change**. ii. On the configuration tab of the sync task, modify the task settings. For more information, see Configure a sync task.
Rename a sync task	To rename a sync task, perform the following steps: i. In the **Sync Tasks** section, move the pointer over the icon next to the sync task that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.

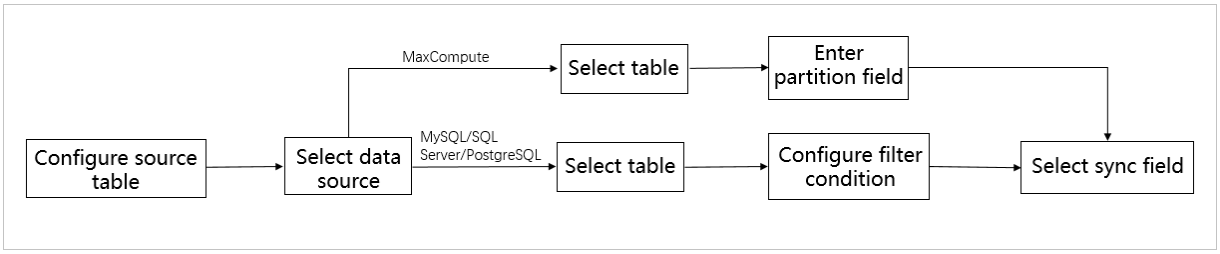
Operation	Description
Move a sync task	To move a sync task, perform the following steps: i. In the **Sync Tasks** section, move the pointer over the icon next to the sync task that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.
Delete a sync task	To delete a sync task, perform the following steps: i. In the **Sync Tasks** section, move the pointer over the icon next to the sync task that you want to delete and select **Delete**. ii. In the **Tip** dialog box, enter your comments. iii. Click **OK**.

9.8.3.4. Configure a sync task

You can configure a sync task to synchronize data from a source table to a destination table. This topic describes how to configure a sync task.

Configuration process

The following figure shows how to configure a sync task.



The following table describes the data sources that Dataphin supports for both the source and destination tables.

Data storage type	Data source
Relational database	MySQL, Vertica, Oracle, SQL Server, PostgreSQL, and PolarDB-X
Analytical database	AnalyticDB, AnalyticDB for MySQL V3.0, and AnalyticDB for PostgreSQL
Alibaba Cloud big data warehouse	MaxCompute
Application database	LogHub
Open-source big data warehouse	Hive and HBase

Data storage type	Data source
Unstructured data storage	FTP, HDFS, and Elasticsearch
NoSQL database	MongoDB

Note If you select HBase, HBase 0.94.x and HBase 1.1.x are supported.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Sync Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.

   You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The Sync Tasks section appears.
3. In the **Sync Tasks** section, click a created sync task. On the tab that appears, set the parameters as required.
   i. Set the parameters in the **Source** section. This section displays parameters based on the data source of the source table that you select. For more information about the parameters in this section, see Data sources.
   ii. Set the parameters in the **Target** section. This section displays parameters based on the data source of the destination table that you select. For more information about the parameters in this section, see Data sources.
   iii. Set the parameters in the **Source Fields** section. After you select a source table in the Source section, the fields of the source table appear in the **Source Fields** section.You can click the icon in the **Actions** column to remove a field.

   If you need to add a field that you have removed, click **Create Field**. In the dialog box that appears, check the **Field and Description** and **Data Type** parameters of the field, select the field, and then click **Add**.
   iv. Set the parameters in the **Target Fields** section. After you select a destination table in the Target section, the fields of the destination table appear in the **Target Fields** section.You can click the icon in the **Actions** column to remove a field.

   If you need to add a field that you have removed, click **Create Field**. In the dialog box that appears, check the **Field and Description** and **Data Type** parameters of the field, select the field, and then click **Add**.

v. Match the source and destination fields.

- If each source field has the same name as a destination field and the number of source fields equals that of destination fields, the system automatically matches the source and destination fields.
- If the source and destination fields have different names, you must manually match the source and destination fields. Remove the destination fields whose names are different from the source fields. Move the pointer over an empty row in the destination field list and select a field from the list that appears.

4. Set required limits for the sync task in the **Sync Limits** section. The parameters in this section are used to control the concurrency and fault tolerance. In most cases, you can use the default settings.

Parameter	Description
Speed Limit	The maximum data transmission rate during synchronization. Default value: 1MB/s. Dataphin will try to reach but will not exceed this rate. This parameter determines the size of scheduled resources when the sync task is running. The higher the speed limit is, the more resources are scheduled.
Concurrent Tasks	The maximum number of concurrent sync tasks.
Error Threshold	The maximum number of errors that are allowed during synchronization before the sync task is terminated. The default value is 0, which indicates that no errors are allowed.

5. Configure the scheduling policy for the sync task. For more information, see Configure a scheduling policy.
6. Save, submit, and then publish the sync task.

i. Click the icon in the upper-right corner to save the sync task.

ii. Click the icon in the upper-right corner to submit the sync task.

iii. (Optional)Publish the sync task.

- If the current project is in Dev mode, publish the sync task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the sync task.

The following table describes the operations that you can perform on a sync task in the **Submitted** state.

Operation	Description

Operation	Description
Modify a sync task	To modify a sync task, perform the following steps: a. In the Sync Tasks section, move the pointer over the icon next to the sync task that you want to modify and select **Change**. b. On the tab that appears, modify the configuration of the sync task. For more information, see Configure a sync task. You can modify a sync task only when it is locked by you or unlocked. If the task is locked by another user, the icon appears in the upper-right corner. You can click the icon to steal the lock. After the lock is stolen, the icon appears, and you can modify the sync task. **Note** After you modify a sync task in the **Submitted** state and save the modification, the status of the sync task changes to **Developing**.
Rename a sync task	To rename a sync task, perform the following steps: a. In the Sync Tasks section, move the pointer over the icon next to the sync task that you want to rename and select **Rename**. b. In the field that appears, enter a new name and press Enter.
Move a sync task	To move a sync task, perform the following steps: a. In the Sync Tasks section, move the pointer over the icon next to the sync task that you want to move and select **Move**. b. In the **Move File** dialog box, set the **Select Directory** parameter to the destination directory and click **OK**.
Unpublish a sync task	To unpublish a sync task, perform the following steps: ■ In the Sync Tasks section, move the pointer over the icon next to the sync task that you want to unpublish and select **Unpublish**. ■ In the **Tip** dialog box, enter your comments and click **OK**.
Unpublish and delete a sync task	To unpublish and delete a sync task, perform the following steps: ■ In the Sync Tasks section, move the pointer over the icon next to the sync task that you want to unpublish and delete and select **Unpublish and Delete**. ■ In the **Tip** dialog box, enter your comments and click **OK**.

9.8.3.5. Configure a scheduling policy

Dataphin allows you to configure the scheduling rules and dependencies of nodes to make sure that tasks can be properly scheduled. This topic describes how to configure a scheduling policy for a sync task.

Prerequisites

Sync tasks are configured. For more information, see Configure a sync task.

Context

- Dataphin allows you to configure scheduling policies only for recurring tasks.
- A dependency is a semantic connection between two or more nodes. The status of an upstream node affects the running of its downstream nodes.
- After you configure dependencies for an upstream node and its downstream nodes, the downstream nodes can be run only after the upstream node is run. Before the system runs a node, the system checks the scheduling time that is configured for the node and determines whether to run the node.
- If a scheduling configuration is submitted before the specified scheduling time, the configuration takes effect after the specified scheduling time. If a dependency is configured after the specified scheduling time, the dependency takes effect for the instances that are generated the next day.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. Go to the **Sync Tasks** section.
   i. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   ii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iii. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The Sync Tasks section appears.
4. In the **Sync Tasks** section, click a configured sync task.
5. On the tab that appears, click **Scheduling Configuration**.
6. In the **Scheduling Configuration** pane, set the parameters as required.

i. Set the parameters in the **Basic Information** section.Dataphin automatically generates the node name, node ID, node type, and owner. You cannot modify these parameters.

Parameter	Description
Description	The description of the scheduling policy.
Priority	The priority based on which the sync task is scheduled. Valid values: ■ **Lowest Priority** ■ **Low Priority** ■ **Medium Priority** ■ **High Priority** ■ **Highest Priority**
Parameters	The specified values for the parameters that are used in the code of the task. Dataphin allows you to specify the parameters that are used in the code of a task. The specified values are used when the task is run. You can click **Parameters and Descriptions** to know how to configure the parameters.

ii. Set the parameters in the **Scheduling Configuration** section.

Parameter	Description
Schedule Mode	The scheduling mode of the task. Valid values: ■ **Normal**: runs the task based on the specified recurrence. By default, this option is selected. ■ **Dry-run**: runs the task based on the specified recurrence. However, the scheduling system does not run the task but returns a success response. ■ **Pause Scheduling**: runs the task based on the specified recurrence. However, the scheduling system does not run the task but returns a failure response. You can select this check box if you need to suspend a task and run it later.

Parameter	Description
Recurrence	The recurrence of the task. Valid values: ■ **Day**: automatically runs the task once per day. When you create a recurring task, the task is set to run at 00:00 every day by default. You can also click the  icon to specify a time for the task to be run as needed. ■ **Week**: automatically runs the task at a specified time point on specified days of each week. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not actually run the instance or consume resources but directly returns a success response. Assume that you set Recurrence to Week and specify that the task is run every Monday and Tuesday. The scheduling system generates and runs instances every Monday and Tuesday. Every Wednesday, Thursday, Friday, Saturday, and Sunday, the scheduling system generates instances and returns success responses without running the instances. ■ **Month**: automatically runs the task at a specified time point on specified days of each month. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not actually run the instance or consume resources but directly returns a success response. Assume that you set Recurrence to Month and specify that the task is run on the seventh day of each month. The scheduling system generates and runs an instance on the seventh day of each month. On the other days, the scheduling system generates instances and returns success responses without running the instances. ■ **Hour**: automatically runs the task at a specified interval during a specified time period or at specified time points every day. The scheduling system automatically generates instances for the task and runs the instances at the specified interval or time points. Assume that you set Recurrence to Hour, select Time Period, and set the Start, End, and Interval parameters to 00:00, 23:59, and 1, respectively. The scheduling system automatically generates instances for the task and runs an instance every hour. ■ **Minute**: automatically runs the task at a specified interval during a specified time period every day. Assume that you set Recurrence to Minute and set the Start, End, and Interval parameters to 00:00, 23:59, and 05, respectively. The scheduling system automatically generates instances for the task and runs an instance every 5 minutes.

Parameter	Description
Depend on Previous Instance	Specifies whether the current task is run after the previous instance of another task or of the current task is run. If you select Depend on Previous Instance, you must further select Current Task or Select Task. ■ If you select **Current Task**, the current task is run after the previous instance of the current task is run. ■ If you select **Select Task**, enter a keyword in the search box that appears to search for and select one or more tasks to depend on.

iii. Set the parameters in the **Dependency** section.

Parameter	Description
Upstream Dependency	The upstream nodes on which the current node depends. To specify an upstream node, perform the following steps: a. Click **Create Upstream Dependency**. b. In the **Create Upstream Dependency** dialog box, search for a node based on the output name. **Note** Each node output name is globally unique in Dataphin. c. Click **OK**. To remove a node from the upstream node list, click the icon in the **Actions** column.
Current Node	The output name for the current node. You can set multiple output names for a node, which can be used to configure dependencies for other nodes. To set an output name, perform the following steps: a. Click **Add**. b. In the **Add Output Task Nodes for Current Task Node** dialog box, enter an output name. Observe uniform rules when you set each output name for the current node. Set an output name in the format of `Project name. Table name`. This helps other users find this node when they configure the upstream dependency for their nodes. c. Click **OK**. You can also perform the following operations on an existing output name: ■ To delete an output name, click the icon in the **Actions** column. ■ To view the downstream nodes after the current node is submitted or published, click the icon in the **Actions** column.

7. Click **OK**.
8. Save, submit, and then publish the sync task.

i. On the configuration tab of the sync task, click the icon in the upper-right corner to save the sync task.

ii. On the configuration tab of the sync task, click the icon in the upper-right corner to submit the sync task.

iii. (Optional)Publish the sync task.

- If the current project is in Dev mode, publish the sync task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the sync task. The task can be scheduled in the production environment after you submit it.

9.8.3.6. Run a one-time sync task

This topic describes how to run a one-time task.

Prerequisites

- One-time tasks are created. For more information, see Create a sync task.
- One-time tasks are configured. For more information, see Configure a sync task.

Context

Description on running of sync tasks:

- You do not need to run recurring tasks. Dataphin automatically runs recurring tasks based on their scheduling settings.
- One-time tasks run only after you trigger them manually.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Prod** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click **Scheduling** in the top navigation bar.
5. On the **Scheduling** page, move the pointer over the left-side navigation submenu and click the icon. The One-time Tasks section appears.
6. Run a one-time task.
   i. In the **One-Time Tasks** section, click the one-time task that you want to run.
   ii. On the page that appears, click **Run** in the upper-right corner.
   iii. In the **Run** dialog box, set the **Instance Name** and **Data Timestamp** parameters based on your business needs. The default value of **Data Timestamp** is the date of the day before.
   iv. Click **OK**.

9.8.3.7. Verify a sync task

This topic describes how to verify a sync task.

Prerequisites

Sync tasks are created and run. For more information about how to run a one-time task, see Run a one-time sync task.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.
5. Create a folder for storing ad hoc query tasks.
   i. In the **Ad Hoc Query** section, click the icon next to **Ad Hoc Query**.
   ii. In the **Create Folder** dialog box, enter a folder name and select a directory.
   iii. Click **OK**.
6. Create an ad hoc query task.
   i. In the **Ad Hoc Query** section, click the icon next to **Ad Hoc Query**.
   ii. In the **Create Item** dialog box, set the **Name**, **Description**, and **Select Directory** parameters.
   iii. Click **OK**. The Code Editor tab appears.
7. Write the code of the ad hoc query task on the Code Editor tab.
   i. Enter the following SQL statement on the Code Editor tab:

```
select * from Destination table name where ds='Data timestamp';
```

   ii. Click **Precompile** in the upper-right corner of the Code Editor tab to check whether the SQL statement complies with the standard.If the SQL statement does not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statement so that it complies with the standard.
   iii. Click **Run** in the upper-right corner of the Code Editor tab to execute the SQL statement. Then, check whether the sync task is running properly based on the result returned by the ad hoc query task.

9.9. Data modeling and development

9.9.1. Data modeling and development

9.9.1.1. Overview

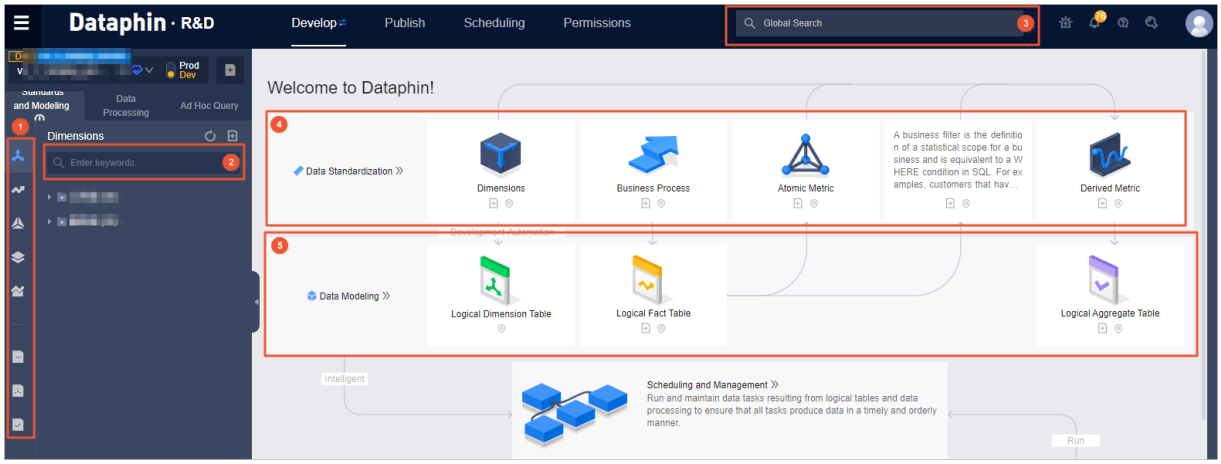
Dataphin provides the data standardization and data modeling features for building data warehouses. These features allow you to build data models in a systematic manner.

Go to the Standards and Modeling tab

To go to the **Standards and Modeling** tab, perform the following steps:

1. Log on to the Dataphin console.
2. Go to the **Standards and Modeling** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

On the **Standards and Modeling** tab, you can create data standardization objects and logical tables.



No.	Description
1	The left-side navigation submenu. Move the pointer over this submenu and click an item as needed. In the section that appears, you can create a corresponding data standardization object or logical table.
2	The local search box. Enter a keyword in the search box to search for data standardization objects or logical tables.
3	The global search box. Enter a keyword in the search box to search for objects in Dataphin. To search for a data distilling object, you must have activated the data distilling feature.
4	The **Data Standardization** section. This section displays the types of data standardization objects. Click the icon below each type to create a corresponding data standardization object.
5	The **Data Modeling** section. This section displays the types of logical tables that are supported for data modeling. Click the icon below each type to create a corresponding logical table.

Prerequisites

- External data is imported to the data sources that are configured for the current project. For more information, see Data ingestion.
- The current project is in Basic or Dev mode. After you go to the Develop page, select a project in the upper-left corner first. Dataphin allows you to use the data standardization and data modeling features for projects in Basic or Dev-Prod mode. For more information, see Create projects and Create business units.
- The current project is bound to a business unit. Otherwise, the data standardization and data modeling features are unavailable. For more information about how to bind a project to a business unit, see Create projects.
- If the current project is in Basic mode, the project is bound to a business unit in Basic mode. If the project in Basic mode is bound to a business unit in Prod mode, you cannot perform data standardization and data modeling. You can only use the data processing and ad hoc query features.
- If the current project is in Dev-Prod mode, the project mode switch is switched to Dev, so that you can create, modify, or delete objects. After you submit a created or modified object, publish the object. After the object is published, it can be automatically scheduled in the corresponding project in Prod mode.

9.9.1.2. Data standardization: Dimensions

9.9.1.2.1. Create a dimension

A dimension reflects a perspective from which you view an object. You may need to view an object from multiple aspects or at multiple levels. To this end, Dataphin allows you to define dimensions for data standardization. This topic describes how to create a dimension.

Prerequisites

Data domains are created. For more information, see Create a data domain.

Context

A dimension is a statistical object. It is an entity that exists, for example, the time, region, and product dimensions. By creating dimensions, you can standardize your business entities or master data during architectural design to guarantee that they are unique.

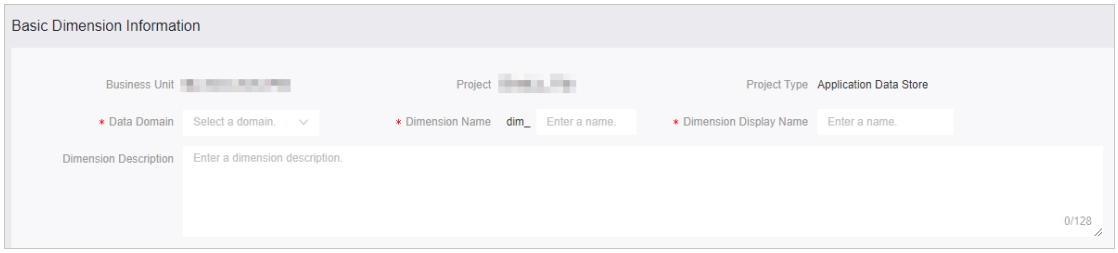
Procedure

1. Log on to the Dataphin console.
2. Go to the **Dimensions** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Dimensions section appears.
3. Go to the **Create Dimension** tab by using one of the following methods:

- In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization > Dimensions**.
- In the **Dimensions** section, click the icon next to **Dimensions**.
- In the right-side workspace of the Develop page, click the icon below **Dimensions**.
- In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Click **Create Dimension**.

4. On the **Create Dimension** tab, set the parameters as required.

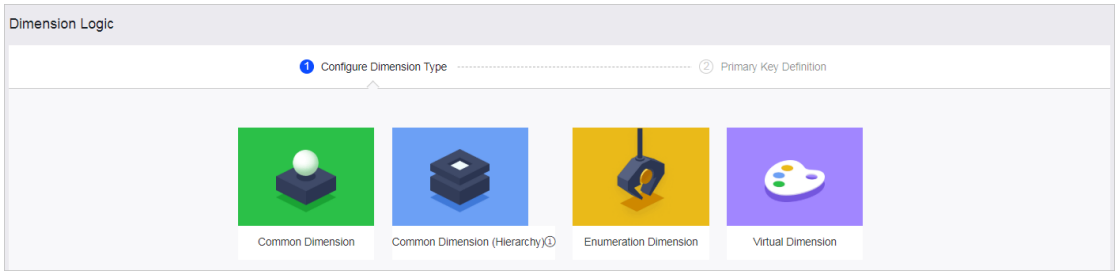
   i. Set the parameters in the **Basic Dimension Information** section.



Parameter	Description
Dimension Name	The name of the dimension. The name can contain letters, digits, and underscores (_). By default, the name starts with the dim_ prefix.
Dimension Display Name	The display name of the dimension. The display name can contain letters, digits, underscores (_), and hyphens (-).
Dimension Description	The description of the dimension to be created.
Data Domain	The data domain to which the dimension belongs. If no data domain is available, create one first.

   ii. Set the parameters in the **Dimension Logic** section.

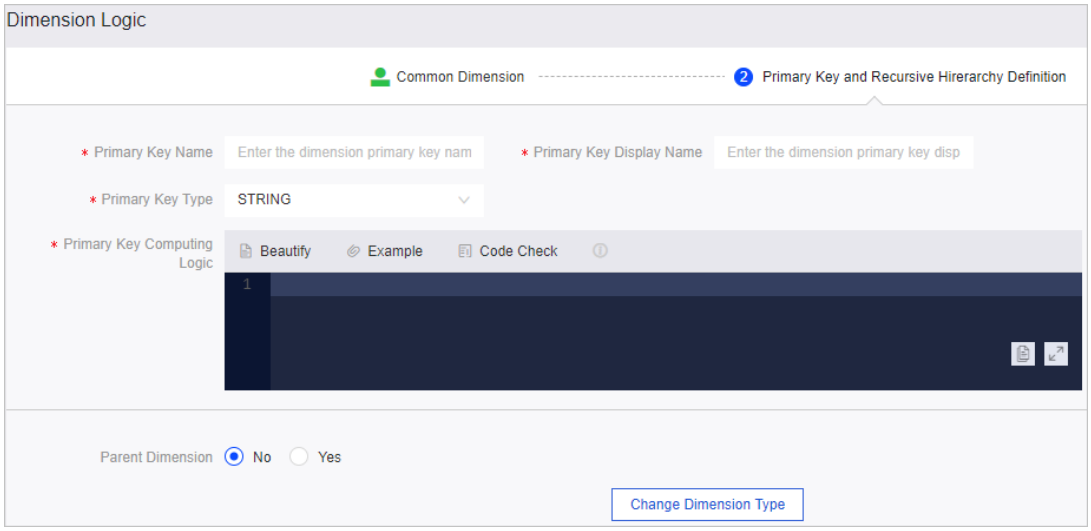
   The dimension logic defines the scope of the dimension and determines the logical characteristics of the dimension. The dimension logic ensures that the dimension is true and unique. The following table describes the four dimension types provided by Dataphin to meet different needs.



Dimension type	Description

Dimension type	Description
Common Dimension	To create a common dimension, enter the primary key information. Set the parent dimension as needed. After you create and publish a common dimension, Dataphin automatically generates a logical dimension table.
Common Dimension (Hierarchy)	To create a common dimension by hierarchy, enter the primary key information and define the dimension hierarchy and logic. You can analyze data from multiple dimensions in a stable hierarchy. After you create and publish a common dimension by hierarchy, Dataphin automatically generates a set of logical dimension tables by hierarchy.
Enumeration Dimension	To define an enumeration dimension, enter enumeration codes and values.
Virtual Dimension	To create a virtual dimension, enter the primary key information. Dataphin allows you to define virtual dimensions that have no business entities and cannot be specified by a fixed data scope in a source table. A virtual dimension can also be used in data standardization and data modeling.

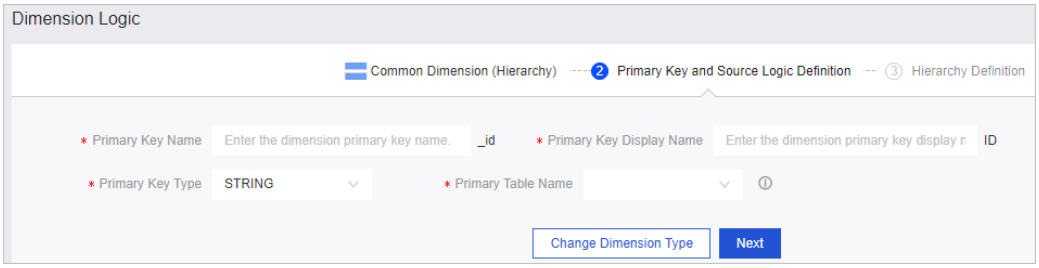
- If you select **Common Dimension**, set the parameters as described in the following table.



Parameter	Description
Primary Key Name	The name of the primary key for the dimension. The name can contain letters, digits, and underscores (_).
Primary Key Display Name	The display name of the primary key for the dimension. The display name can contain letters, digits, underscores (_), and hyphens (-).
Primary Key Type	The data type of the primary key for the dimension. Valid values: **STRING**, **BIGINT**, **DOUBLE**, **DATETIME**, and **DECIMAL**.
Primary Key Computing Logic	The computing logic of the primary key for the dimension. To prevent maintenance exceptions or data errors, configure the time-based partitioning conditions in the code. a. Click **Example** to view the description about the computing logic of the primary key for the dimension and the sample SQL statement. `select province from dataphin_test where ds='${bizdate}';` b. After you enter an SQL statement, click **Code Check** to verify whether the SQL statement you entered is valid.
Parent Dimension	The parent dimension of the dimension. Valid values: **Yes** and **No**. If you select **Yes**, select a parent dimension from the drop-down list.

- If you select **Common Dimension (Hierarchy)**, perform the following steps to set the parameters:

a. Set the parameters in the **Primary Key and Source Logic Definition** step and click **Next**.

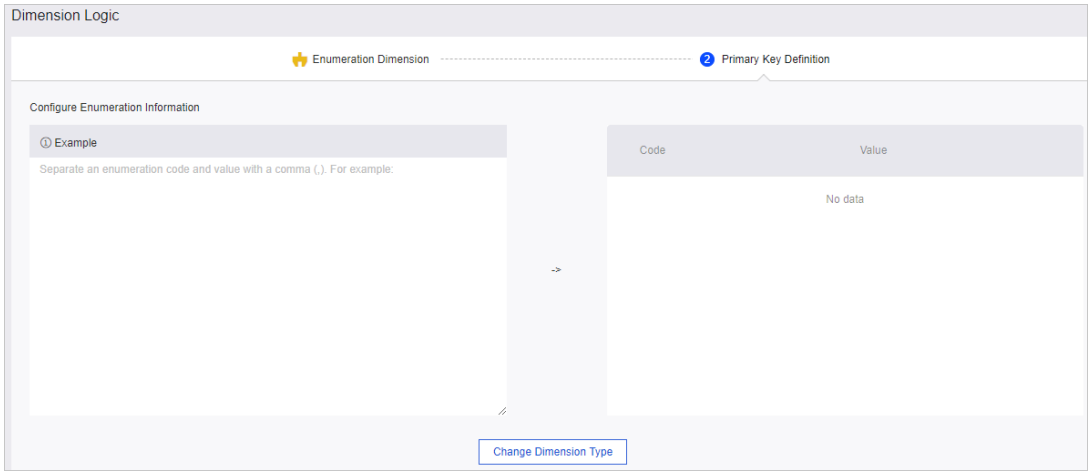


Parameter	Description
Primary Key Name	The name of the primary key for the dimension. The name can contain letters, digits, and underscores (_).
Primary Key Display Name	The display name of the primary key for the dimension. The display name can contain letters, digits, underscores (_), and hyphens (-).
Primary Key Type	The data type of the primary key for the dimension. Valid values: **STRING**, **BIGINT**, **DOUBLE**, **DATETIME**, and **DECIMAL**.
Primary Table Name	The name of the main source table for the dimension. This table is also the destination table for a sync task. **Note** We recommend that you select a table from the production environment.

b. Set the parameters in the **Hierarchy Definition** step.

Parameter	Description
Child Field	The child field in the hierarchy.
Parent Field	The parent field in the hierarchy.
Name Field	The field of the dimension name in the hierarchy.
Number of Levels	The number of levels in the hierarchy. A maximum of nine levels are supported.
Generate Leaf Dimension	Specifies whether to generate a leaf dimension. Valid values: ■ **Yes**: Dataphin automatically generates a leaf dimension. ■ **No**: Dataphin generates a leaf dimension only if you have defined the computing logic for the leaf dimension.
Root-level Dimension Definition	The judgment condition for the root-level dimension based on the field in the main source table. For example, you can enter is_parent='ture', where parent indicates the field in the main source table.
Time-based Partitioning	The judgment condition for the data timestamp and statistical period based on the field in the main source table. For example, you can enter ds=${bizdate}.

■ If you select **Enumeration Dimension**, set the parameters as instructed.



In the left-side field of the **Configure Enumeration Information** section, enter enumeration codes and values. Separate each enumeration code and its value with a comma (,).

```
F,female
M,male
```

In the right-side field, Dataphin automatically parses the enumeration codes and values that you entered in the **Code** and **Value** columns, respectively.

- If you select **Virtual Dimension**, set the parameters as described in the following table.

Parameter	Description
Primary Key Name	The name of the primary key for the dimension. The name can contain letters, digits, and underscores (_).
Primary Key Display Name	The display name of the primary key for the dimension. The display name can contain letters, digits, underscores (_), and hyphens (-).
Primary Key Type	The data type of the primary key for the dimension. Valid values: **STRING**, **BIGINT**, **DOUBLE**, **DATETIME**, and **DECIMAL**.

> **Note** If a virtual dimension is associated with a logical dimension table or a logical fact table, you can create derived metrics based on this virtual dimension. The value of the virtual dimension used to compose the statistic granularity is extracted from the associated field.

5. After you set the parameters on the **Create Dimension** tab, click the icon in the upper-right corner to save the dimension.
6. Submit the dimension.
   i. On the **Create Dimension** tab, click the icon in the upper-right corner to submit the dimension.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
   - After you submit a common dimension, Dataphin automatically generates the corresponding logical dimension table for the dimension.
   - After you submit a common dimension by hierarchy, Dataphin generates a set of logical dimension tables by hierarchy for the dimension.
7. (Optional)Publish the dimension.
   - If the current project is in Dev mode, publish the dimension to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the dimension after you submit it.

9.9.1.2.2. View the logical dimension table of a dimension

This topic describes how to view the logical dimension table that is generated for a dimension.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Dimensions** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Dimensions section appears.
3. View the logical dimension table of a dimension by using one of the following methods:
   - In the **Dimensions** section, click the data domain to which the dimension for which you want to view the logical dimension table belongs and the type of the dimension. Click the dimension. On the View Attributes tab, move the pointer over the icon and select **Logical Dimension Table**.
   - In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Find the dimension for which you want to view the logical dimension table and click the icon in the **Actions** column.
4. On the tab that appears, view the information about the logical dimension table of the dimension.

9.9.1.2.3. Modify a dimension

A dimension reflects a perspective from which you view an object. You may need to view an object from multiple aspects or at multiple levels. To this end, Dataphin allows you to define dimensions for data standardization. This topic describes how to modify a dimension.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Dimensions** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Dimensions section appears.
3. Go to the **Change Dimension** tab by using one of the following methods:
   - In the **Dimensions** section, click the data domain to which the dimension that you want to modify belongs and the type of the dimension. Click the dimension. On the View Attributes

tab, click **Change**.

- In the **Dimensions** section, click the data domain to which the dimension that you want to modify belongs and the type of the dimension. Move the pointer over the icon next to the dimension and select **Change**.
- In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Find the dimension that you want to modify and click the icon in the **Actions** column.

4. (Optional)Steal the lock of the dimension.
   - If the dimension is locked by yourself, you do not need to steal the lock.
   - If the dimension is locked by another user, click the icon in the upper-right corner of the **Change Dimension** tab to steal the lock.
5. On the **Change Dimension** tab, modify the parameters as needed. For more information, see Create a dimension.
6. After you modify the parameters on the **Change Dimension** tab, click the icon in the upper-right corner to save the dimension.
7. Submit the dimension.
   i. On the **Change Dimension** tab, click the icon in the upper-right corner to submit the dimension.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
8. (Optional)Publish the dimension.
   - If the current project is in Dev mode, publish the dimension to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the dimension after you submit it.

9.9.1.2.4. Unpublish and delete a dimension

This topic describes how to unpublish, unpublish and delete, and delete dimensions in different states.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Context

- A dimension may be in the following states:
  - After you create and save a dimension, it enters the **Draft** state.
  - After you submit a dimension, it enters the **Submitted** state.
  - After you modify and save a dimension in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a dimension in the **Submitted** state, it enters the **Draft** state.

- You can unpublish only the dimensions in the **Developing** or **Submitted** state.
- You can delete only the dimensions in the **Draft** state.
- You can unpublish and delete only the dimensions in the **Developing** or **Submitted** state.

Unpublish a dimension

> **Note** You can unpublish only a dimension for which no dependency is configured.

1. Log on to the Dataphin console.
2. Go to the **Dimensions** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Dimensions section appears.
3. (Optional)Steal the lock of the dimension to be unpublished.
   - If the dimension is locked by yourself, you do not need to steal the lock.
   - If the dimension is locked by another user, click the icon in the upper-right corner of the **Change Dimension** tab to steal the lock. For more information about how to go to the **Change Dimension** tab, see Modify a dimension.
4. Open the **Tip** dialog box for unpublishing a dimension by using one of the following methods:
   - In the **Dimensions** section, click the data domain to which the dimension that you want to unpublish belongs and the type of the dimension. Click the dimension. On the View Attributes tab, move the pointer over the icon and select **Unpublish**.
   - In the **Dimensions** section, click the data domain to which the dimension that you want to unpublish belongs and the type of the dimension. Move the pointer over the icon next to the dimension and select **Unpublish**.
   - In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Find the dimension that you want to unpublish and choose > **Unpublish** in the **Actions** column.
5. In the **Tip** dialog box, enter your comments.
6. Click **OK**.

Delete a dimension

1. (Optional)Steal the lock of the dimension to be deleted.
   - If the dimension is locked by yourself, you do not need to steal the lock.
   - If the dimension is locked by another user, click the icon in the upper-right corner of the

Change Dimension tab to steal the lock. For more information about how to go to the **Change Dimension** tab, see Modify a dimension.

2. Open the **Tip** dialog box for deleting a dimension by using one of the following methods:
   - In the **Dimensions** section, click the data domain to which the dimension that you want to delete belongs and the type of the dimension. Move the pointer over the icon next to the dimension and select **Delete**.
   - In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Find the dimension that you want to delete and choose > **Delete** in the **Actions** column.
3. In the **Tip** dialog box, enter your comments.
4. Click **OK**.

Unpublish and delete a dimension

> **Note** You can unpublish and delete only a dimension for which no dependency is configured.

1. (Optional)Steal the lock of the dimension to be unpublished and deleted.
   - If the dimension is locked by yourself, you do not need to steal the lock.
   - If the dimension is locked by another user, click the icon in the upper-right corner of the **Change Dimension** tab to steal the lock. For more information about how to go to the **Change Dimension** tab, see Modify a dimension.
2. Open the **Tip** dialog box for unpublishing and deleting a dimension by using one of the following methods:
   - In the **Dimensions** section, click the data domain to which the dimension that you want to unpublish and delete belongs and the type of the dimension. Move the pointer over the icon next to the dimension and select **Unpublish and Delete**.
   - In the **Dimensions** section, click **Dimensions Object List** in the lower part. The **Dimensions** tab of the **Object Explorer** tab appears. Find the dimension that you want to unpublish and delete and choose > **Unpublish and Delete** in the **Actions** column.
   - In the **Dimensions** section, click the data domain to which the dimension that you want to unpublish and delete belongs and the type of the dimension. Click the dimension. On the View Attributes tab, move the pointer over the icon and select **Unpublish and Delete**.
3. In the **Tip** dialog box, enter your comments.
4. Click **OK**.

9.9.1.3. Data standardization: Business processes

9.9.1.3.1. Create a business process

A business process describes the events that are necessary for specific business. For example, a business process in commerce involves placing orders, paying, and refunding. By creating a business process, you can standardize the events in specific business during architectural design and make sure that they are unique. This topic describes how to create a business process.

Prerequisites

Data domains are created. For more information, see Create a data domain.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Processes section appears.
3. Open the **Create Business Process** dialog box by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization** > **Business Processes**.
   - In the Business Processes section, click the icon next to **Business Processes**.
   - In the right-side workspace of the Develop page, click the icon below **Business Process**.
   - In the Business Processes section, click **Business Processes Object List** in the lower part. The **Business Processes** tab of the **Object Explorer** tab appears. Click **Create Business Process**.
4. In the **Create Business Process** dialog box, set the parameters as required.

Parameter	Description
Data Domain	The data domain to which the business process to be created belongs.
Name	The name of the business process. The name can contain letters, digits, and underscores (_).
Display Name	The display name of the business process. The display name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the business process.

5. Submit the business process.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.

iii. Click **OK**.

6. (Optional)Publish the business process.
   - If the current project is in Dev mode, publish the business process to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business process after you submit it.

9.9.1.3.2. Modify a business process

A business process describes the events that are necessary for specific business. For example, a business process in commerce involves placing orders, paying, and refunding. This topic describes how to modify a business process.

Prerequisites

Business processes are created. For more information, see Create a business process.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the  icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the  icon. The Business Processes section appears.
3. Open the **Change Business Process** dialog box by using one of the following methods:
   - In the **Business Processes** section, click the data domain to which the business process that you want to modify belongs. Click the business process. On the View Attributes tab, click **Change**.
   - In the **Business Processes** section, click the data domain to which the business process that you want to modify belongs. Move the pointer over the  icon next to the business process and select **Change**.
   - In the **Business Processes** section, click **Business Processes Object List** in the lower part. The **Business Processes** tab of the **Object Explorer** tab appears. Find the business process that you want to modify and click the  icon in the **Actions** column.
4. In the **Change Business Process** dialog box, modify the parameters as needed. For more information, see Create a business process.
5. Submit the business process.
   i. After you modify the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.

6. (Optional)Publish the business process.
   - If the current project is in Dev mode, publish the business process to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business process after you submit it.

9.9.1.3.3. Clone a business process

This topic describes how to clone a business process.

Prerequisites

Business processes are created. For more information, see Create a business process.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Processes section appears.
3. Open the **Create Business Process** dialog box by using one of the following methods:
   - In the **Business Processes** section, click the data domain to which the business process that you want to clone belongs. Click the business process. On the View Attributes tab, move the pointer over the icon and select **Clone**.
   - In the **Business Processes** section, click **Business Processes Object List** in the lower part. The **Business Processes** tab of the **Object Explorer** tab appears. Find the business process that you want to clone and choose > **Clone** in the **Actions** column.
4. In the **Create Business Process** dialog box, set the parameters as required. For more information, see Create a business process.
5. Submit the business process.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. (Optional)Publish the business process.
   - If the current project is in Dev mode, publish the business process to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business process after you submit it.

9.9.1.3.4. Create a logical table

This topic describes how to create a logical fact table based on a business process.

Prerequisites

Business processes are created. For more information, see Create a business process.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Processes section appears.
3. Open the **Create Logical Fact Table** dialog box by using one of the following methods:
   - In the **Business Processes** section, click the data domain to which the business process for which you want to create a logical fact table belongs. Click the business process. On the View Attributes tab, move the pointer over the icon and select **Create Logical Table**.
   - In the **Business Processes** section, click **Business Processes Object List** in the lower part. The **Business Processes** tab of the **Object Explorer** tab appears. Find the business process for which you want to create a logical fact table and click the icon in the **Actions** column.
4. In the **Create Logical Fact Table** dialog box, set the parameters as required. For more information, see Create a logical fact table.
5. Submit and publish the logical fact table that you create. For more information, see Create a logical fact table.

9.9.1.3.5. View logical fact tables related to a business process

This topic describes how to view logical fact tables that are related to a business process.

Prerequisites

Business processes are created. For more information, see Create a business process.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.

i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Processes section appears.

3. In the **Business Processes** section, click **Business Processes Object List** in the lower part.
4. On the **Business Processes** tab of the **Object Explorer** tab, find the business process for which you want to view related logical fact tables and click the icon in the **Actions** column.
5. In the **Related Logical Fact Tables** list, view the logical fact tables that are related to the business process.

9.9.1.3.6. Delete a business process

A business process describes the events that are necessary for specific business. For example, a business process in commerce involves placing orders, paying, and refunding. This topic describes how to delete a business process.

Prerequisites

Business processes are created. For more information, see Create a business process.

Context

You can delete only the business processes that have no related logical fact tables.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Processes** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Processes section appears.
3. Open the **Tip** dialog box for deleting a business process by using one of the following methods:
   - In the **Business Processes** section, click the data domain to which the business process that you want to delete belongs. Click the business process. On the View Attributes tab, move the pointer over the icon and select **Delete**.

- In the **Business Processes** section, click the data domain to which the business process that you want to delete belongs. Move the pointer over the icon next to the business process and select **Delete**.
- In the **Business Processes** section, click **Business Processes Object List** in the lower part. The **Business Processes** tab of the **Object Explorer** tab appears. Find the business process that you want to delete and choose > **Delete** in the **Actions** column.

4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

9.9.1.4. Logical tables: Logical dimension tables

9.9.1.4.1. Edit model information

One logical dimension table corresponds to one dimension. A logical dimension table stores dimension attributes that describe facts. After you submit or publish a dimension, Dataphin automatically generates a logical dimension table. For the logical dimension table, Dataphin allows you to add attributes, create a dimension association, add child dimensions, and configure a logical table conversion task.

Usage notes

You can edit a logical dimension table by creating a dimension association, adding attributes, and adding child dimensions. Note the following points when you edit model information:

- You only need to store persistent attributes of a dimension in its corresponding logical dimension table.
- You do not need to define fields for child dimensions if these fields are defined in the published parent dimension. Dataphin can automatically reference such fields.

Create a dimension association

1. Log on to the Dataphin console.
2. Go to the **Logical Dimension Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Dimension Tables section appears.
3. In the **Logical Dimension Tables** section, click the logical dimension table for which you want to create a dimension association. The configuration tab of the logical dimension table appears.
4. On the configuration tab of the logical dimension table, click **Create Dimension Association** in the Central Table section.
5. In the **Create Dimension Association** dialog box, set the parameters as required.

Section	Description
Associated Dimension	The dimension to be associated with the current logical dimension table.
Edit Association Logic	The logic for associating a field in the current logical dimension table with the specified dimension. Select a field from the drop-down list under Dimension Table Field to Be Associated.
Edit Dimension Role	The role of the specified dimension. In this section, you must specify the **Role Name** and **Role Display Name** parameters.
Default Value Settings	The value of the specified field when the field cannot be associated with the specified dimension. By default, the Default Value parameter is set to -110.

6. Click **OK**.

Add attributes

1. In the **Logical Dimension Tables** section, click the logical dimension table for which you want to add attributes. The configuration tab of the logical dimension table appears.
2. On the configuration tab of the logical dimension table, click **Add Attributes** in the Central Table section.
3. In the **Create Attribute** dialog box, set the parameters as required.
   - If you set the **Source Table** parameter to **Import Fields**, perform the following steps to add fields: Click the icon next to each desired field in the **Select New Fields** section to add the field to the **Create Field** section. Then, enter a display name for the field in the **Field Display Name** column. Click **Save and Verify**.
   - If you set the **Source Table** parameter to **Customize Fields**, enter SQL statements in the code editor to define fields. You can click **Example** in the upper part of the code editor to view the sample SQL statement. Click **Code Check** to verify the syntax of the SQL statements you entered. After the SQL statements pass the verification, click **Save and Verify**.

Create a child dimension

1. In the **Logical Dimension Tables** section, click the logical dimension table for which you want to create a child dimension. The configuration tab of the logical dimension table appears.
2. On the configuration tab of the logical dimension table, click **Add Child Dimension** in the Central Table section.
3. On the **Inherit from Dimension** tab, set the parameters as required.

Section	Parameter	Description
Basic Dimension Information	**Dimension Name**	The name of the child dimension.
	Dimension Display Name	The display name of the child dimension.

Section	Parameter	Description
	Dimension Description	The description of the child dimension.
Dimension Logic	Primary Key Name	The name of the primary key of the child dimension.
	Primary Key Display Name	The display name of the primary key of the child dimension.
	Primary Key Type	The data type of the primary key of the child dimension. Click the icon and select a data type from the drop-down list.
	Primary Key Computing Logic	The primary key computing logic that you want to define for the child dimension. Perform the following steps: i. Click **Example** to view the description about the primary key computing logic and the sample SQL statement. ii. In the code editor, enter SQL statements for defining the primary key computing logic. iii. Click **Code Check** to verify the syntax of the SQL statements you entered.
	Parent Dimension	The parent dimension of the child dimension to be created. By default, the dimension that corresponds to the current logical dimension table is selected as the parent dimension. You can also click the icon and select a parent dimension from the drop-down list.

4. On the **Inherit from Dimension** tab, click the icon in the upper-right corner to save the inherited dimension.
5. Submit the inherited dimension.
   i. On the **Inherit from Dimension** tab, click the icon in the upper-right corner to submit the inherited dimension.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. Publish the inherited dimension.
   - If the current project is in Dev mode, publish the inherited dimension to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the inherited dimension after you submit it.

9.9.1.4.2. Edit a central table

This topic describes how to edit a central table for a logical dimension table, such as adding attributes, creating dimension associations, and defining computing logic.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Add attributes

1. Log on to the Dataphin console.
2. Go to the **Logical Dimension Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Dimension Tables section appears.
3. In the **Logical Dimension Tables** section, click the logical dimension table to which you want to add attributes. The configuration tab of the logical dimension table appears.
4. (Optional)Steal the lock of the logical dimension table.
   - If the logical dimension table is locked by yourself, you do not need to steal the lock.
   - If the logical dimension table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
6. On the **Central Table Settings** tab, move the pointer over **Create Field** and select **Attribute**.
7. In the **Create Attribute** dialog box, set the parameters as required. For more information, see Add attributes.
8. Save, submit, and then publish the logical dimension table.
   i. After you add attributes, click the icon in the upper-right corner of the configuration tab to save the attributes.
   ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.

v. (Optional)Publish the logical dimension table.

- If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

Create a dimension association

1. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
2. On the **Central Table Settings** tab, move the pointer over **Create Field** and select **Dimension Association**.
3. In the **Create Dimension Association** dialog box, set the parameters as required. For more information, see Create a dimension association.
4. Save, submit, and then publish the logical dimension table.
   i. After you create a dimension association, click the icon in the upper-right corner of the configuration tab to save the dimension association.
   ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
   v. (Optional)Publish the logical dimension table.
   - If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

Define public computing logic

1. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
2. On the **Central Table Settings** tab, click **Define Public Computing Logic** next to **Create Field**.
3. In the **Define Public Computing Logic** dialog box, read the description on the right and click **Create Public Computing Logic** on the left.
4. In the dialog box that appears, enter a name for the public computing logic in the field and click **OK**. A code editor appears.
5. Perform the following steps to define the public computing logic:
   i. Click **Example** to view the description about the public computing logic and the sample SQL statement.
   ii. Enter SQL statements.

iii. Click **Code Check** to verify the SQL statements you entered.

If the SQL statements do not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statements so that they comply with the standard.

iv. After the SQL statements pass the verification, click **Save**.

You can also perform the following steps to delete existing computing logic:

- Select the computing logic to be deleted or click **Select All** in the lower part of the Define Public Computing Logic dialog box.
- Click **Delete**.
- In the **Delete Public Computing Logic** message, click **OK**.
- Another **Delete Public Computing Logic** message appears. Click **OK**.

6. Click **Finish** in the lower-right corner of the Define Public Computing Logic dialog box.
7. Save, submit, and then publish the logical dimension table.

i. After you define the computing logic, click the icon in the upper-right corner of the configuration tab to save the computing logic.

ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.

iii. In the dialog box that appears, enter your comments.

iv. Click **OK**.

v. (Optional)Publish the logical dimension table.

- If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

View computing logic

1. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
2. On the **Central Table Settings** tab, click **Group by Type** to view the primary key, attributes, associated dimensions, and system field of the logical dimension table. Click **Group by Source** to view the primary key, system field, and source logical tables of the logical dimension table.
3. On the **Group by Type** or **Group by Source** tab, find the field for which you want to view the computing logic and click the icon in the **Computing Logic (All)** column.

You can use one of the following methods to filter fields:

- Click the icon next to **Computing Logic (All)** to filter fields by computing logic status.
- Enter a keyword in the search box to search for fields.

Modify a field

> **Note** On the configuration tab of a logical dimension table, you can only modify fields that are associated with dimensions.

If you need to modify the primary key, move the pointer over the icon in the **Actions** column and click **Details**. On the tab that appears, modify the parameters as needed. For more information, see Modify a dimension.

To modify fields that are associated with dimensions, perform the following steps:

1. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
2. On the **Central Table Settings** tab, click **Group by Type** or **Group by Source** to view fields as needed.
3. On the **Group by Type** or **Group by Source** tab, find the field that you want to modify and click the icon in the **Actions** column.
4. In the **Change Field** dialog box, modify the **Field Name**, **Field Display Name**, **Description**, and **Data Type** parameters as needed.
5. Click **OK**.
6. In the **Confirm Close** message, click **OK**.
7. Save, submit, and then publish the logical dimension table.
   i. After you modify the field, click the icon in the upper-right corner of the configuration tab to save the modified field.
   ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
   v. (Optional)Publish the logical dimension table.
      - If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

Delete a field

> **Note** You can delete only the fields that are associated with dimensions.

1. On the configuration tab of the logical dimension table, click **Central Table Settings** in the top navigation bar.
2. On the **Central Table Settings** tab, click **Group by Type** or **Group by Source** to view fields as needed.
3. On the **Group by Type** or **Group by Source** tab, find the field that you want to delete and click the icon in the **Actions** column.

4. In the message that appears, click **OK**.

9.9.1.4.3. Configure a logical table conversion task

To specify the retention period, partition fields, and custom parameters for a logical dimension table, you can configure a logical table conversion task for the table. This topic describes how to configure a logical table conversion task for a logical dimension table.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Dimension Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Dimension Tables section appears.
3. In the **Logical Dimension Tables** section, click the logical dimension table for which you want to configure a logical table conversion task. The configuration tab of the logical dimension table appears.
4. (Optional)Steal the lock of the logical dimension table.
   - If the logical dimension table is locked by yourself, you do not need to steal the lock.
   - If the logical dimension table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical dimension table, click **Logical Table Conversion Task Settings** in the top navigation bar.
6. In the **Logical Table Conversion Task Settings** pane, set the parameters as required.

Parameter	Description
Retention (Days)	The retention period of the logical dimension table. You can enter a value in the field or select one of the following options: 7, 14, 30, and 365.
Select Partitioning Fields	The partition fields of the logical dimension table.

Parameter	Description
Custom Parameter Settings	The custom parameters to be specified for the logical dimension table.

7. Click **OK**.
8. Save, submit, and then publish the logical dimension table.
   i. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to save the logical table conversion task.
   ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
   v. (Optional)Publish the logical dimension table.
      - If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

9.9.1.4.4. Configure a scheduling policy

Dataphin allows you to configure the scheduling rules and dependencies of nodes to make sure that tasks can be properly scheduled. This topic describes how to configure a scheduling policy for a logical dimension table.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Dimension Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Dimension Tables section appears.
3. In the **Logical Dimension Tables** section, click the logical dimension table for which you want

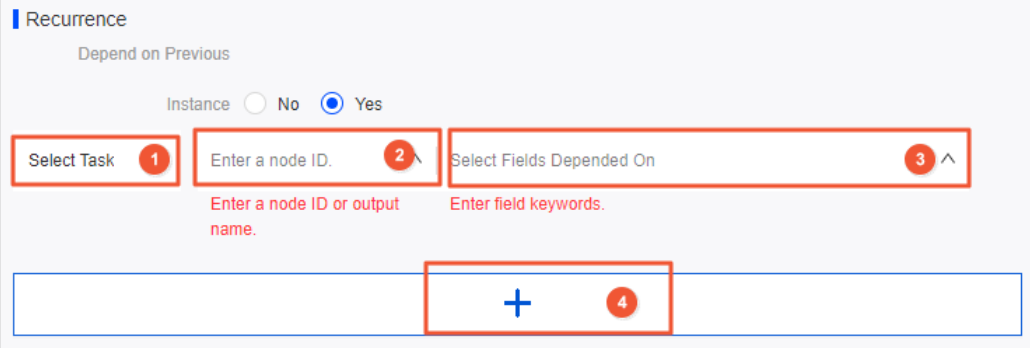
to configure a scheduling policy. The configuration tab of the logical dimension table appears.

4. (Optional)Steal the lock of the logical dimension table.
   - If the logical dimension table is locked by yourself, you do not need to steal the lock.
   - If the logical dimension table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical dimension table, click **Scheduling Configuration** in the top navigation bar.
6. In the **Scheduling Configuration** pane, set the parameters as required.
   i. Set the parameters in the **Basic Information** section.Dataphin automatically generates values for the **Logical Table Name**, **Task Type**, and **Description** parameters. You cannot change these values.

Parameter	Description
Description	The description of the scheduling policy.
Priority	The priority based on which the logical table task is scheduled. Valid values: ■ **Lowest Priority** ■ **Low Priority** ■ **Medium Priority** ■ **High Priority** ■ **Highest Priority**
Parameters	The specified values for the parameters that are used in the code of the task. You can click **Parameters and Descriptions** to view the rules for setting task parameters and the time parameters that are supported in the scheduling system of Dataphin.

   ii. Set the Depend on Previous Instance parameter in the **Recurrence** section.
   - If you set the **Depend on Previous Instance** parameter to **No**, the task of the logical dimension table does not depend on previous instances.
   - If you set the **Depend on Previous Instance** parameter to **Yes**, perform the following steps to configure dependencies:
     a. Click the + icon.

b. Set the parameters as required.



No.	Description
1	Select the node type. You can select **Select Task** or **Nodes of This Logical Table** from the drop-down list.
2	■ If you select **Select Task**, specify a node ID by using one of the following methods: ■ Click the icon, enter a keyword to search for the desired node ID, and then select the node ID. ■ Click Enter a node ID and select a node ID from the drop-down list. ■ If you select **Nodes of This Logical Table**, click the icon, select the fields to depend on, and then click **OK**.
3	Select the fields to depend on by using one of the following methods: ■ Click the icon, enter a keyword to search for fields, select the desired fields, and then click OK. ■ Click Select Fields Depended On, select the desired fields, and then click **OK**.
4	To add more dependencies, click the icon.

iii. Set the parameters in the **Dependency** section.

Parameter or section	Description
Automatic Parse	Specifies whether to automatically parse the dependencies of the logical table task. If you click **Parse Input and Output**, Dataphin automatically parses the task dependencies on instances of upstream nodes and the current node, and displays the dependency information in the **Upstream Dependency** and **Logical Table Node (This Node)** sections, respectively. If you click **Parse Input and Output** repeatedly, the system updates the information in the **Upstream Dependency** and **Logical Table Node (This Node)** sections.
Upstream Dependency	The upstream nodes on which the physical node of the logical dimension table depends. You can add a dependent upstream node for the logical table task by using one of the following methods: ■ Add a parsed node as a dependent upstream node. a. Click **Parse Error Information**. Source physical tables and other nodes related to the logical dimension table appear. b. Select a node and click **Confirm Association**. ■ Add a system node, for example, a zero-load node, as a dependent upstream node. a. Click **Add Upstream Dependency**. b. Select an upstream node and a source physical table. You can click the + icon to add more dependent upstream nodes. c. Click **OK**.
Logical Table Node (This Node)	The output name of the logical table task. Dataphin automatically generates the output information about the task.

7. Click **OK**.
8. Save, submit, and then publish the logical dimension table.
   i. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to save the configured scheduling policy.
   ii. On the configuration tab of the logical dimension table, click the icon in the upper-right corner to submit the logical dimension table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.

v. (Optional)Publish the logical dimension table.

- If the current project is in Dev mode, publish the logical dimension table to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the logical dimension table after you submit it.

9.9.1.4.5. Unpublish and delete a logical dimension table

This topic describes how to unpublish and delete logical dimension tables in different states.

Prerequisites

Dimensions are created. For more information, see Create a dimension.

Context

A logical dimension table may be in the following states:

- After you create and save a logical dimension table, it enters the **Draft** state.
- After you submit a logical dimension table, it enters the **Submitted** state.
- After you modify and save a logical dimension table in the **Submitted** state, it enters the **Developing** state.
- After you unpublish a logical dimension table in the **Submitted** state, it enters the **Draft** state.

Limits on unpublishing and deleting logical dimension tables:

- You can unpublish only the logical dimension tables in the **Developing** or **Submitted** state.
- You can delete only the logical dimension tables in the **Draft** state.

Unpublish a logical dimension table

1. Log on to the Dataphin console.
2. Go to the **Logical Dimension Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project. You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Dimension Tables section appears.
3. In the **Logical Dimension Tables** section, move the pointer over the icon next to the logical dimension table that you want to unpublish and select **Unpublish**.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Delete a logical dimension table

1. In the **Logical Dimension Tables** section, move the pointer over the icon next to the logical dimension table that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.1.5. Logical tables: Logical fact tables

9.9.1.5.1. Create a logical fact table

A logical fact table describes a business process in detail. This topic describes how to create a logical fact table.

Prerequisites

Data domains are created.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.
3. Open the **Create Logical Fact Table** dialog box by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Logical Tables > Logical Fact Tables**.
   - In the **Logical Fact Tables** section, click the icon next to **Logical Fact Tables**.
   - In the right-side workspace of the Develop page, click the icon below **Logical Fact Table**.
4. In the **Create Logical Fact Table** dialog box, perform the following steps:

i. Set the parameters in the **Basic Information** step.

Parameter	Description
Data Domain	The data domain to which the business process to be described by the logical fact table belongs.
Business Process	The business process to be described by the logical fact table.
Fact Table Type	The type of the logical fact table. Valid values: **Transaction Fact Table** and **Periodic Snapshot Fact Table**. ■ Transaction fact tables track and measure events that occur in business activities. For example, you can design a transaction fact table to track and measure the order payment event in order transactions. ■ Periodic snapshot fact tables measure the status of entities at regular intervals. For example, you can use a periodic snapshot fact table to track the account balance or commodity inventory.
Name	The name of the logical fact table. The value of the Name parameter is in the `fct_Business process name_Custom name_di` or `fct_Business process name_Custom name_df` format. The suffix of the name varies with the value of the Fact Table Type parameter. ■ If you set the Fact Table Type parameter to **Transaction Fact Table**, Dataphin automatically suffixes the name of the logical fact table with di. ■ If you set the Fact Table Type parameter to **Periodic Snapshot Fact Table**, Dataphin automatically suffixes the name of the logical fact table with df.
Display Name	The display name of the logical fact table. The display name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the logical fact table.
Main Source Table	The main source table of the logical fact table. You can select a physical table or a logical table as the main source table.

ii. Click **Next**.

iii. Set the parameters in the **Primary Key Definition** step.

- If you set the **Set Primary Key** parameter to **No**, you can determine whether to set the Filter Conditions parameter based on your needs.
- If you set the **Set Primary Key** parameter to **Yes**, set the required parameters as described in the following table.

Parameter	Description
Field Name	The name of the primary key field. Example: item_id
Field Display Name	The display name of the primary key field. Example: commodity_id
Field Data Type	The data type of the primary key field.
Field Computing Logic	The computing logic of the primary key field.

> **Note** If you specify a unique primary key field, for example, order_id, for a logical fact table, you must associate other fields that are added to the logical fact table with the primary key field.

iv. Click **Submit**.

5. Add measures.

i. On the configuration tab of the logical fact table, open the **Create Measure** dialog box by using one of the following methods:

- Click **Add Measure** in the **Central Table** section.
- Click **Central Table Settings** in the top navigation bar. On the Central Table Settings tab, move the pointer over **Create Field** and select **Measure**.

ii. In the **Create Measure** dialog box, set the parameters as required.

- If you set the **Source Table** parameter to **Import Fields**, perform the following steps to add fields: Click the icon next to each desired field in the **Select New Fields** section to add the field to the **Create Field** section. Then, enter a display name for the field in the **Field Display Name** column. Click **Save and Verify**.
- If you set the **Source Table** parameter to **Customize Fields**, enter SQL statements in the code editor to define fields. You can click **Example** in the upper part of the code editor to view the sample SQL statement. Click **Code Check** to verify the syntax of the SQL statements you entered. After the SQL statements pass the verification, click **Save and Verify**.

6. Add dimension associations.

i. On the configuration tab of the logical fact table, open the **Create Dimension Association** dialog box by using one of the following methods:

- Click **Create Dimension Association** in the **Central Table** section.
- Click **Central Table Settings** in the top navigation bar. On the Central Table Settings tab, move the pointer over **Create Field** and select **Dimension Association**.

ii. In the **Create Dimension Association** dialog box, set the parameters as required.

Parameter	Description
Associated Dimension	The dimension to be associated with the current logical fact table.
Edit Association Logic	The logic for associating a field in the current logical fact table with the specified dimension. Select a measure from the drop-down list under Fact Table Field to Be Associated.
Edit Dimension Role	The role of the specified dimension. ■ **Role Name**: the role name of the dimension. By default, Dataphin sets this parameter to the name of the specified dimension. ■ **Role Display Name**: the role display name of the dimension. By default, Dataphin sets this parameter to the display name of the specified dimension. You can change the values of the **Role Name** and **Role Display Name** parameters based on your business needs.
Default Value Settings	The default value that is used when the foreign key of the logical fact table cannot be associated with the specified dimension. By default, the **Default Value** parameter is set to -110.

7. Add fact attributes.
   i. On the configuration tab of the logical fact table, open the **Create Fact Attribute** dialog box by using one of the following methods:
      - Click **Add Fact Attribute** in the **Central Table** section.
      - Click **Central Table Settings** in the top navigation bar. On the Central Table Settings tab, move the pointer over **Create Field** and select **Fact Attribute**.
   ii. In the **Create Fact Attribute** dialog box, set the parameters as required.
      - If you set the **Source Table** parameter to **Import Fields**, perform the following steps to add fields: Click the icon next to each desired field in the **Select New Fields** section to add the field to the **Create Field** section. Then, enter a display name for the field in the **Field Display Name** column. Click **Save and Verify**.
      - If you set the **Source Table** parameter to **Customize Fields**, enter SQL statements in the code editor to define fields. You can click **Example** in the upper part of the code editor to view the sample SQL statement. Click **Code Check** to verify the syntax of the SQL statements you entered. After the SQL statements pass the verification, click **Save and Verify**.
8. Submit and publish the logical fact table.
   i. On the configuration tab of the logical fact table, click the **Submit** icon in the upper-right corner to submit the logical fact table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.

iv. (Optional)Publish the logical fact table.

- If the current project is in Dev mode, publish the logical fact table to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the logical fact table after you submit it.

9.9.1.5.2. Configure a logical table conversion task

To specify the retention period, partition fields, and custom parameters for a logical fact table, you can configure a logical table conversion task for the table. This topic describes how to configure a logical table conversion task for a logical fact table.

Prerequisites

Logical fact tables are created. For more information, see Create a logical fact table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.
3. In the **Logical Fact Tables** section, move the pointer over the icon next to the logical fact table for which you want to configure a logical table conversion task and select **Change**.
4. (Optional)Steal the lock of the logical fact table.
   - If the logical fact table is locked by yourself, you do not need to steal the lock.
   - If the logical fact table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical fact table, click **Logical Table Conversion Task Settings** in the top navigation bar.
6. In the **Logical Table Conversion Task Settings** pane, set the parameters as required.

Parameter	Description
Retention (Days)	The retention period of the logical fact table. You can enter a value in the field or select one of the following options: 7, 14, 30, and 365.

Parameter	Description
Select Partitioning Fields	The partition fields of the logical fact table.
Custom Parameter Settings	The custom parameters to be specified for the logical fact table.

7. Click **OK**.
8. Save, submit, and then publish the logical fact table.
   i. On the configuration tab of the logical fact table, click the icon in the upper-right corner to save the logical table conversion task.
   ii. On the configuration tab of the logical fact table, click the icon in the upper-right corner to submit the logical fact table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
   v. (Optional)Publish the logical fact table.
      - If the current project is in Dev mode, publish the logical fact table to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the logical fact table after you submit it.

9.9.1.5.3. Modify a logical fact table

A logical fact table describes a business process in detail. This topic describes how to modify a logical fact table.

Prerequisites

Logical fact tables are created. For more information, see Create a logical fact table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.

3. In the **Logical Fact Tables** section, move the pointer over the icon next to the logical fact table that you want to modify and select **Change**.
4. (Optional)Steal the lock of the logical fact table.
   - If the logical fact table is locked by yourself, you do not need to steal the lock.
   - If the logical fact table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical fact table, modify settings about the central table, logical table conversion task, and scheduling policy. For more information, see Create a logical fact table, Configure a scheduling policy, and Configure a logical table conversion task.
6. Save, submit, and then publish the logical fact table.
   i. On the configuration tab of the logical fact table, click the icon in the upper-right corner to save the logical fact table.
   ii. On the configuration tab of the logical fact table, click the icon in the upper-right corner to submit the logical fact table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
   v. (Optional)Publish the logical fact table.
      - If the current project is in Dev mode, publish the logical fact table to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the logical fact table after you submit it.

9.9.1.5.4. Configure a scheduling policy

Dataphin allows you to configure the scheduling rules and dependencies of nodes to make sure that tasks can be properly scheduled. This topic describes how to configure a scheduling policy for a logical fact table.

Prerequisites

Logical fact tables are created. For more information, see Create a logical fact table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

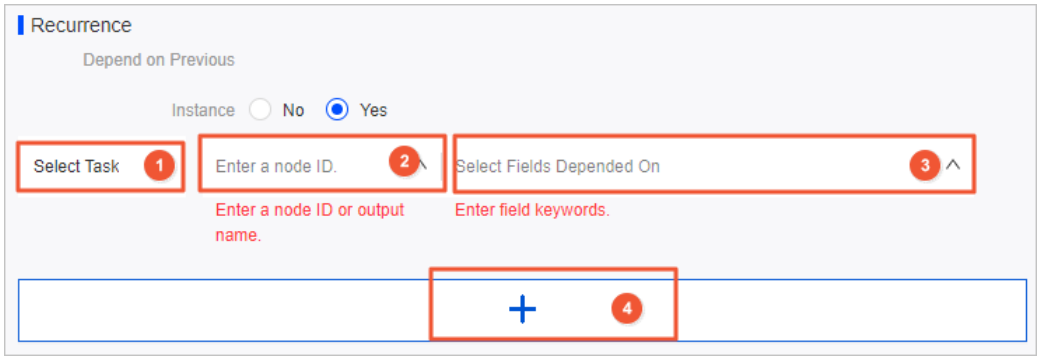
iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.

3. In the **Logical Fact Tables** section, click the logical fact table for which you want to configure a scheduling policy. The configuration tab of the logical fact table appears.
4. (Optional)Steal the lock of the logical fact table.
   - If the logical fact table is locked by yourself, you do not need to steal the lock.
   - If the logical fact table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical fact table, click **Scheduling Configuration** in the top navigation bar.
6. In the **Scheduling Configuration** pane, set the parameters as required.
   i. Set the parameters in the **Basic Information** section.Dataphin automatically generates values for the **Logical Table Name**, **Task Type**, and **Description** parameters. You cannot change these values.

Parameter	Description
Description	The description of the scheduling policy.
Priority	The priority based on which the logical table task is scheduled. Valid values: ■ **Lowest Priority** ■ **Low Priority** ■ **Medium Priority** ■ **High Priority** ■ **Highest Priority**
Parameters	The specified values for the parameters that are used in the code of the task. You can click **Parameters and Descriptions** to view the rules for setting task parameters and the time parameters that are supported in the scheduling system of Dataphin.

   ii. Set the Depend on Previous Instance parameter in the **Recurrence** section.
   - If you set the **Depend on Previous Instance** parameter to **No**, the task of the logical fact table does not depend on previous instances.

- If you set the **Depend on Previous Instance** parameter to **Yes**, perform the following steps to configure dependencies:
  a. Click the + icon.
  b. Set the parameters as required.



No.	Description
1	Select the node type. You can select **Select Task** or **Nodes of This Logical Table** from the drop-down list.
2	■ If you select **Select Task**, specify a node ID by using one of the following methods: ■ Click the icon, enter a keyword to search for the desired node ID, and then select the node ID. ■ Click Enter a node ID and select a node ID from the drop-down list. ■ If you select **Nodes of This Logical Table**, click the icon, select the fields to depend on, and then click **OK**.
3	Select the fields to depend on by using one of the following methods: ■ Click the icon, enter a keyword to search for fields, select the desired fields, and then click **OK**. ■ Click Select Fields Depended On, select the desired fields, and then click **OK**.
4	To add more dependencies, click the + icon.

iii. Set the parameters in the **Dependency** section.

Parameter or section	Description
Automatic Parse	Specifies whether to automatically parse the dependencies of the logical table task. If you click **Parse Input and Output**, Dataphin automatically parses the task dependencies on instances of upstream nodes and the current node, and displays the dependency information in the **Upstream Dependency** and **Logical Table Node (This Node)** sections, respectively. If you click **Parse Input and Output** repeatedly, the system updates the information in the **Upstream Dependency** and **Logical Table Node (This Node)** sections.
Upstream Dependency	The upstream nodes on which the physical node of the logical fact table depends. You can add a dependent upstream node for the logical table task by using one of the following methods: ■ Add a parsed node as a dependent upstream node. a. Click **Parse Error Information**. Source physical tables and other nodes related to the logical fact table appear. b. Select a node and click **Confirm Association**. ■ Add a system node, for example, a zero-load node, as a dependent upstream node. a. Click **Add Upstream Dependency**. b. Select an upstream node and a source physical table. You can click the + icon to add more dependent upstream nodes. c. Click **OK**.
Logical Table Node (This Node)	The output name of the logical table task. Dataphin automatically generates the output information about the task.

7. Click **OK**.
8. Save, submit, and then publish the logical fact table.
   i. On the configuration tab of the logical fact table, click the icon in the upper-right corner to save the configured scheduling policy.
   ii. On the configuration tab of the logical fact table, click the icon in the upper-right corner to submit the logical fact table.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.

v. (Optional)Publish the logical fact table.
   - If the current project is in Dev mode, publish the logical fact table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical fact table after you submit it.

9.9.1.5.5. View historical version information

This topic describes how to view historical version information about a logical fact table.

Prerequisites

Logical fact tables are created. For more information, see Create a logical fact table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.
3. In the **Logical Fact Tables** section, move the pointer over the icon next to the logical fact table for which you want to view historical version information and select **Change**.
4. On the configuration tab of the logical fact table, click **Versions** in the top navigation bar.
5. In the **Versions** pane, view the historical version information about the logical fact table.
6. Find a version and click **Details** in the **Actions** column.
7. On the page that appears, you can view the basic information, scheduling information, and code details about the version. You can also view the settings of the logical table conversion task in this version.

9.9.1.5.6. Unpublish and delete a logical fact table

This topic describes how to unpublish and delete logical fact tables in different states.

Prerequisites

Logical fact tables are created. For more information, see Create a logical fact table.

Context

A logical fact table may be in the following states:

- After you create and save a logical fact table, it enters the **Draft** state.
- After you submit a logical fact table, it enters the **Submitted** state.
- After you modify and save a logical fact table in the **Submitted** state, it enters the **Developing** state.
- After you unpublish a logical fact table in the **Submitted** state, it enters the **Draft** state.

Limits on unpublishing and deleting logical fact tables:

- You can unpublish only the logical fact tables in the **Developing** or **Submitted** state.
- You can delete only the logical fact tables in the **Draft** state.

Unpublish a logical fact table

1. Log on to the Dataphin console.
2. Go to the **Logical Fact Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project. You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Fact Tables section appears.
3. In the **Logical Fact Tables** section, move the pointer over the icon next to the logical fact table that you want to unpublish and select **Unpublish**.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Delete a logical fact table

1. In the **Logical Fact Tables** section, move the pointer over the icon next to the logical fact table that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.1.6. Data standardization: Atomic metrics

9.9.1.6.1. Create an atomic metric

An atomic metric is an abstraction of statistical criteria and specific algorithms. Dataphin introduces the concept of development automation. To define a metric, you must specify its statistical criteria, which is the computing logic. This increases development efficiency and ensures the consistency of statistical results.

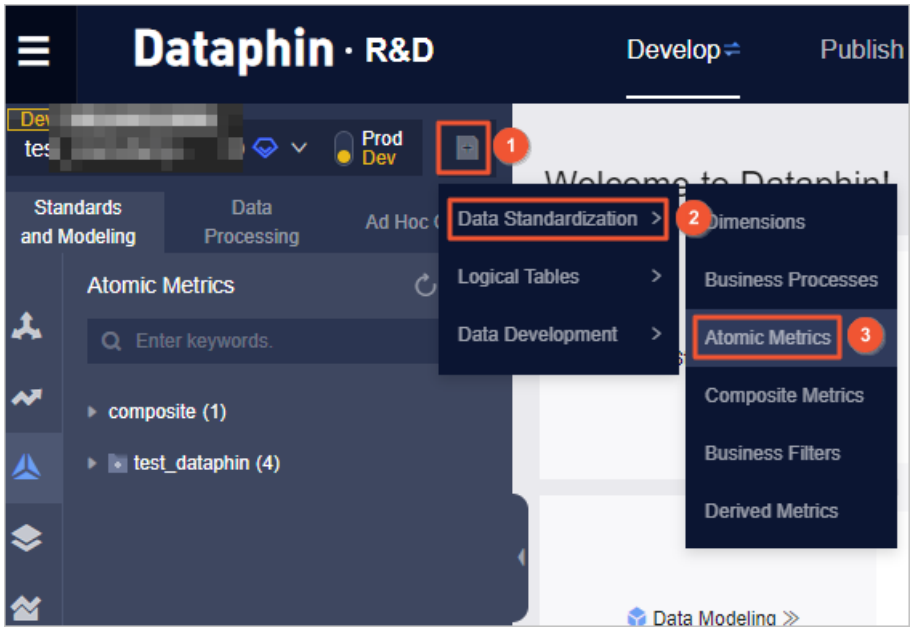
Context

Dataphin allows you to create atomic metrics and composite metrics based on different computing logic. Before you create an atomic metric or a composite metric, take note of the following descriptions:

- An atomic metric is a metric whose value is directly obtained, for example, a payment amount.
- A composite metric is a metric whose value is calculated based on submitted atomic metrics and specified computing logic. For example, you have submitted two atomic metrics. One atomic metric measures the total payment amount and the other measures the number of customers who made the payments. Then, you can create a composite metric that uses the two atomic metrics to calculate the average payment amount per customer.

Create an atomic metric

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. Go to the **Create Atomic Metric** tab by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization > Atomic Metrics**.

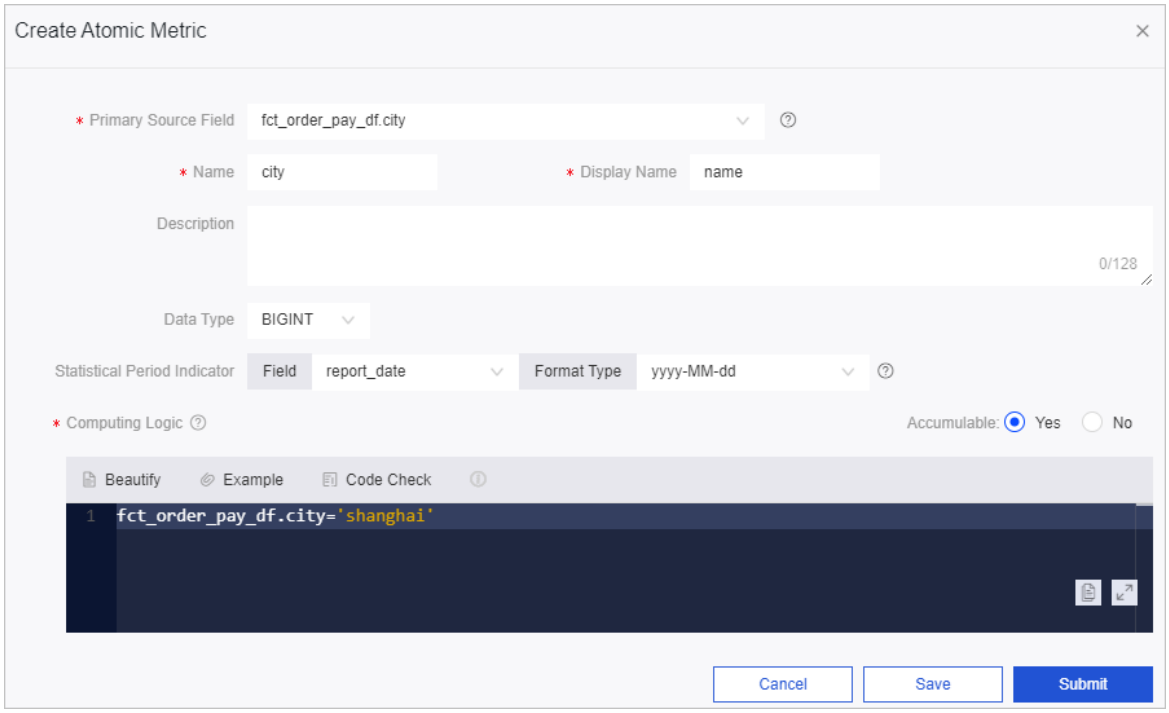


- In the Atomic Metrics section, click the icon next to **Atomic Metrics** and select **Create Atomic Metric.**
- In the right-side workspace of the Develop page, click the icon below **Atomic Metric** and select **Create Atomic Metric.**
- In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears. Click **Create Atomic Metric** in the upper-right corner and select **Atomic Metrics.**

4. On the **Create Atomic Metric** tab, select a data domain and a source table in the **Select Source Information** section.

Parameter	Description
Data Domain	The data domain of the business process to which the source table that you want to select belongs.
Source Table	The source table that contains the fields that you want to reference in the computing logic of the atomic metric. To ensure that all atomic metric models are standard, Dataphin supports only logical dimension tables and logical fact tables as the source table. **Note** After you select the source table, a list of existing atomic metrics that are defined based on the source table appears in the Atomic Metrics section.

5. In the **Atomic Metrics** section, click **Create Atomic Metric.**
6. In the **Create Atomic Metric** dialog box, set the parameters as required.



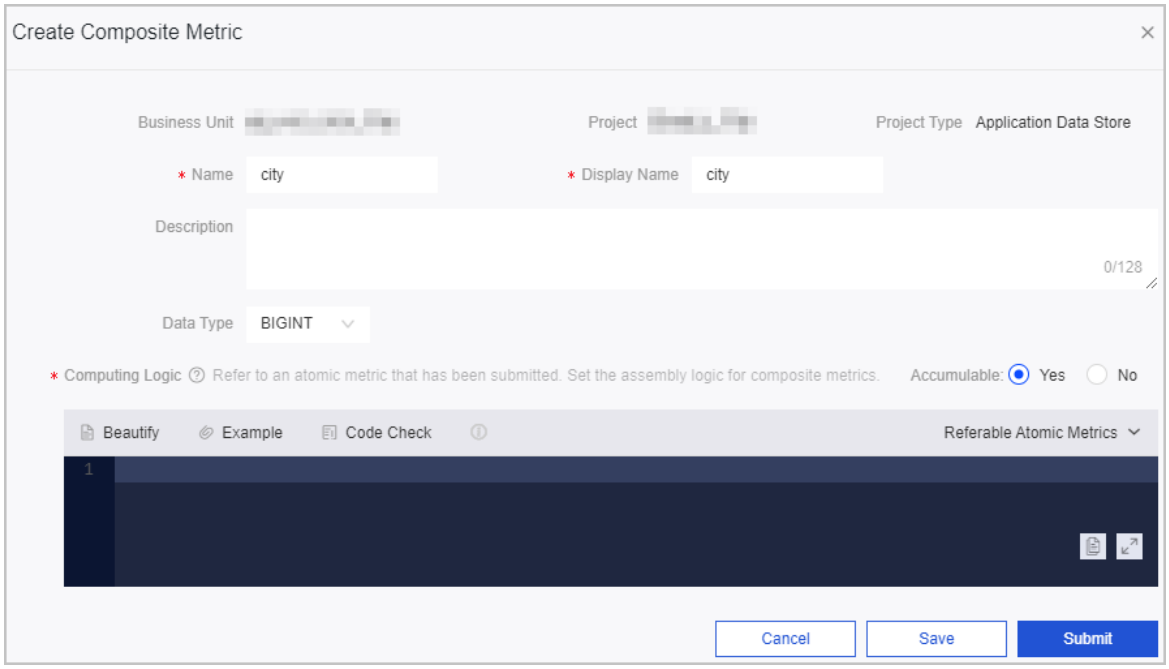
Parameter	Description
Primary Source Field	The primary source field of the atomic metric. You can select Entire Table or a field in the source table from the drop-down list.
Name	The name of the atomic metric. The name of the selected primary source field is automatically entered in the Name field. You can modify the name as needed.
Display Name	The display name of the atomic metric. The display name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the atomic metric.
Data Type	The data type of the atomic metric. Valid values: **STRING**, **BIGINT**, **DOUBLE**, **DATETIME**, and **DECIMAL**.
Field	The field to be associated with the statistical period. By default, the time-based partition field in the business unit to which the atomic metric belongs is selected.
Format Type	The format of the value of the field associated with the statistical period when the value is used in computation.

Parameter	Description
Computing Logic	The computing logic of the atomic metric. Enter an SQL statement, for example, count(distinct order_id), to define the atomic metric based on the source logical table. In the SQL statement, order_id is a field in the source table. Perform the following steps to define the computing logic: i. Click **Example** to view the description about the computing logic of atomic metrics and the sample SQL statement. ii. After you enter an SQL statement, click **Code Check** to verify the SQL statement you entered.

7. Submit and publish the atomic metric.
   i. After you set the parameters, click **Submit**.
   ii. (Optional)Publish the atomic metric.
   - If the current project is in Dev mode, publish the atomic metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the atomic metric. The atomic metric can be used for scheduling after you submit it.

Create a composite metric

1. Open the **Create Composite Metric** dialog box by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization > Composite Metrics**.
   - In the **Atomic Metrics** section, click the icon next to Atomic Metrics and select **Create Composite Metric**.
   - In the right-side workspace of the Develop page, click the icon below **Atomic Metric** and select **Create Composite Metric**.
   - In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears. Click **Create Atomic Metric** in the upper-right corner and select **Composite**.
2. In the **Create Composite Metric** dialog box, set the parameters as required.



Parameter	Description
Name	The name of the composite metric. The name can contain letters, digits, and underscores (_).
Display Name	The display name of the composite metric. The display name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the composite metric.
Data Type	The data type of the composite metric. Valid values: **STRING**, **BIGINT**, **DOUBLE**, **DATETIME**, and **DECIMAL**.

Parameter	Description
Computing Logic	The computing logic of the composite metric. When you define the computing logic, you can use submitted atomic metrics as needed. ○ In the code editor, enter an SQL statement in the form of a formula that uses submitted atomic metrics to define the computing logic of the composite metric. ○ a. Click the icon next to **Referable Atomic Metrics** and select a data domain and a source table. b. In the atomic metric list that appears, click **Add** next to an atomic metric to reference the atomic metric in the SQL statement you entered. You can also enter a keyword in the search box next to the icon to search for atomic metrics. In the atomic metric list that appears, click **Add** next to an atomic metric to reference the atomic metric in the SQL statement you entered. For example, crt_amt and usr_cnt are two submitted atomic metrics that measure the total payment amount and the number of customers who made the payments, respectively. You can enter `crt_amt/usr_cnt` in the code editor to define a composite metric that calculates the average payment amount per customer.

3. After you set the parameters, click **Submit**.
4. (Optional)Publish the composite metric.
   - If the current project is in Dev mode, publish the composite metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the composite metric. The composite metric can be used for scheduling after you submit it.

9.9.1.6.2. Modify an atomic metric

An atomic metric is an abstraction of statistical criteria and specific algorithms. To define an atomic metric, you must specify its statistical criteria, which is the computing logic. This topic describes how to modify an atomic metric.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Context

After you modify the computing logic of an atomic metric, the modification applies to all the derived metrics that are created based on the atomic metric. Before you modify an atomic metric, you must consider the impact on the related derived metrics.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. Open the **Change Atomic Metric** dialog box by using one of the following methods:
   - In the Atomic Metrics section, move the pointer over the icon next to the atomic metric that you want to modify and select **Change**.
   - In the Atomic Metrics section, click the atomic metric that you want to modify. On the View Attributes tab, click **Change**.
   - a. In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears.
     b. Find the atomic metric that you want to modify and click the icon in the **Actions** column.
4. In the **Change Atomic Metric** dialog box, modify the parameters as needed. For more information, see Create an atomic metric.
5. After you modify the parameters, click **Submit**.
6. (Optional)Publish the atomic metric.
   - If the current project is in Dev mode, publish the atomic metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the atomic metric. The atomic metric can be used for scheduling after you submit it.

9.9.1.6.3. View atomic metrics that share the same source table

Dataphin allows you to create multiple atomic metrics based on the same source logical table. This topic describes how to view atomic metrics that share the same source table.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Context

Dataphin allows you to view the atomic metrics that share the same source table with a specific atomic metric.

> **Note** Composite metrics do not share the same source table with each other.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. In the **Atomic Metrics** section, click the atomic metric that want to view. On the View Attributes tab, click **View Same Source**.
4. On the Same-Source Atomic Metrics tab, view the atomic metrics that share the same source table with the atomic metric.

9.9.1.6.4. Clone an atomic metric

Dataphin allows you to clone an atomic metric. This operation creates an atomic metric that shares the same source table with an existing atomic metric. This topic describes how to clone an atomic metric.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. In the **Atomic Metrics** section, click the atomic metric that you want to clone. On the View Attributes tab, move the pointer over the icon and select **Clone**.
4. In the **Clone Atomic Metric** dialog box, set the parameters as required. For more information,

see Create an atomic metric.

5. Click **Submit**.
6. In the dialog box that appears, enter your comments.
7. Click **OK**.
8. (Optional)Publish the atomic metric.
   - If the current project is in Dev mode, publish the atomic metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the atomic metric. The atomic metric can be used for scheduling after you submit it.

9.9.1.6.5. View derived metrics created based on an atomic metric

Dataphin allows you to view derived metrics that are created based on an atomic metric. This topic describes how to view derived metrics of an atomic metric.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Context

Dataphin allows you to create derived metrics based on atomic metrics in the **Submitted** and Developing states. You can view derived metrics of only the atomic metrics in the **Submitted** and Developing states.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. In the **Atomic Metrics** section, click the atomic metric for which you want to view derived metrics. On the View Attributes tab, move the pointer over the icon and select **Related Derived Metrics**.
4. On the Derived Metrics tab of the **Object Explorer** tab, view all the derived metrics that are created based on the atomic metric.

9.9.1.6.6. Create a derived metric

Dataphin allows you to create a derived metric based on an atomic metric. This topic describes how to create a derived metric based on an atomic metric.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. In the **Atomic Metrics** section, click the atomic metric based on which you want to create a derived metric. On the View Attributes tab, move the pointer over the icon and select **Create Derived Metric**.
4. On the **Create Derived Metric** tab, set the parameters as required. For more information, see Create a derived metric.
5. Submit the derived metric.
   i. After you set the parameters, click **Preview Derived Metric**.
   ii. Click **Submit**.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
6. (Optional)Publish the derived metric.
   - If the current project is in Dev mode, publish the derived metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the derived metric. The derived metric can be used for scheduling after you submit it.

9.9.1.6.7. Unpublish and delete an atomic metric

This topic describes how to unpublish, unpublish and delete, and delete atomic metrics in different states.

Prerequisites

Atomic metrics are created. For more information, see Create an atomic metric.

Context

- An atomic metric may be in the following states:
  - After you create and save an atomic metric, it enters the **Draft** state.
  - After you submit an atomic metric, it enters the **Submitted** state.
  - After you modify and save an atomic metric in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish an atomic metric in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing atomic metrics:
  - You can unpublish only the atomic metrics in the **Developing** or **Submitted** state.
  - Only the super administrator, project administrator, and developer can unpublish atomic metrics.
  - Before you unpublish an atomic metric, make sure that the atomic metric is not used by any derived metrics or other atomic metrics.
- Limits on deleting atomic metrics:
  - You can delete only the atomic metrics in the **Draft** state.
  - Only the super administrator, project administrator, and developer can delete atomic metrics.
- Limits on unpublishing and deleting atomic metrics:
  - You can unpublish and delete only the atomic metrics in the **Developing** or **Submitted** state.
  - Only the super administrator, project administrator, and developer can unpublish and delete atomic metrics.
  - Before you unpublish and delete an atomic metric, make sure that the atomic metric is not used by any derived metrics or other atomic metrics.

Delete an atomic metric

1. Log on to the Dataphin console.
2. Go to the **Atomic Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the Standards and Modeling tab, move the pointer over the left-side navigation submenu and click the icon. The Atomic Metrics section appears.
3. Open the **Tip** dialog box for deleting an atomic metric by using one of the following methods:
   - In the Atomic Metrics section, move the pointer over the icon next to the atomic metric

that you want to delete and select **Delete**.

- In the Atomic Metrics section, click the atomic metric that you want to delete. On the View Attributes tab, move the pointer over the icon and select **Delete**.
- a. In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears.
  b. Find the atomic metric that you want to delete and choose > **Delete** in the **Actions** column.

4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Unpublish an atomic metric

1. Open the **Tip** dialog box for unpublishing an atomic metric by using one of the following methods:
   - In the Atomic Metrics section, move the pointer over the icon next to the atomic metric that you want to unpublish and select **Unpublish**.
   - In the Atomic Metrics section, click the atomic metric that you want to unpublish. On the View Attributes tab, move the pointer over the icon and select **Unpublish**.
   - a. In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears.
     b. Find the atomic metric that you want to unpublish and choose > **Unpublish** in the **Actions** column.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

Unpublish and delete an atomic metric

1. Open the **Tip** dialog box for unpublishing and deleting an atomic metric by using one of the following methods:
   - In the Atomic Metrics section, move the pointer over the icon next to the atomic metric that you want to unpublish and delete and select **Unpublish and Delete**.
   - In the Atomic Metrics section, click the atomic metric that you want to unpublish and delete. On the View Attributes tab, move the pointer over the icon and select **Unpublish and Delete**.
   - a. In the Atomic Metrics section, click **Atomic Metrics Object List** in the lower part. The **Atomic Metrics** tab of the **Object Explorer** tab appears.
     b. Find the atomic metric that you want to unpublish and delete and choose > **Unpublish and Delete** in the **Actions** column.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.1.7. Data standardization: Business filters

9.9.1.7.1. Create a business filter

A business filter is used to define the scope of business to be measured. This topic describes how to create a business filter.

Prerequisites

Data domains are created. For more information, see Create a data domain.

Context

- Dataphin allows you to create business filters based on logical dimension tables or logical fact tables.
- Business filters are used to define the conditions for derived metrics in a standardized manner, whereas atomic metrics are used to define the computing logic of derived metrics in a standardized manner.
- To make sure that derived metrics can be created in a uniform and standard manner, you must create business filters for a business unit based on the same logical table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Filters** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.
3. Go to the **Create Business Filter** tab by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization** > **Business Filters**.
   - In the **Business Filters** section, click the icon next to **Business Filters**.
   - In the right-side workspace of the Develop page, click the icon below **Business Filter**.
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Click **Create Business Filter**.
4. On the **Create Business Filter** tab, set the **Data Domain** and **Source Table** parameters in the Select Source Information section.
5. In the **Business Filter Source** section, click **Create Business Filter**.

6. In the **Create Business Filter** dialog box, set the parameters as required.

Parameter	Description
Primary Source Field	The primary source field of the business filter.
Name	The name of the business filter. The name can contain letters, digits, and underscores (_).
Display Name	The display name of the business filter. The display name can contain letters, digits, underscores (_), and hyphens (-).
Description	The description of the business filter.
Computing Logic	The computing logic of the business filter. i. Click **Example** to view the description about the computing logic of business filters and the sample SQL statement. `select province from dataphin_test where ds='${bizdate}';` ii. Click **Code Check** to verify the SQL statements you entered.

7. Submit the business filter.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
8. (Optional)Publish the business filter.
   - If the current project is in Dev mode, publish the business filter to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business filter after you submit it.

9.9.1.7.2. Clone a business filter

This topic describes how to clone a business filter.

Prerequisites

Business filters are created. For more information, see Create a business filter.

Context

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Filters** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.

3. Open the **Clone Business Filter** dialog box by using one of the following methods:
   - In the **Business Filters** section, click the data domain to which the business filter that you want to clone belongs. Click the business filter. On the View Attributes tab, move the pointer over the icon and select **Clone**.
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter that you want to clone and choose > **Clone** in the **Actions** column.
4. In the **Clone Business Filter** dialog box, set the parameters as required.
5. Submit the business filter.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. (Optional)Publish the business filter.
   - If the current project is in Dev mode, publish the business filter to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business filter after you submit it.

9.9.1.7.3. View business filters that share the same source table

Dataphin allows you to create multiple business filters based on the same source logical table. This topic describes how to view business filters that share the same source table.

view business filters that share the same source table

Prerequisites

Business filters are created. For more information, see Create a business filter.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Filters** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project. You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.

3. Go to the **Same-Source Business Filters** tab by using one of the following methods:
   - In the **Business Filters** section, click the data domain to which the business filter for which you want to view business filters that share the same source table belongs. Click the business filter. On the View Attributes tab, click **View Same Source**.
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter for which you want to view business filters that share the same source table and click the icon in the **Actions** column.
4. On the **Same-Source Business Filters** tab, view the business filters that share the same source table with the business filter.

9.9.1.7.4. View derived metrics created based on a business filter

Dataphin allows you to view derived metrics that are created based on a business filter. This topic describes how to view derived metrics of a business filter.

Prerequisites

Business filters are created. For more information, see Create a business filter.

Context

Dataphin allows you to create derived metrics based on business filters in the **Submitted** and Developing states. You can view derived metrics of only the business filters in the **Submitted** and Developing states.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Filters** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.

3. Go to the **Derived Metrics** tab by using one of the following methods:
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter for which you want to view derived metrics and click the icon in the **Actions** column.
   - In the **Business Filters** section, click the data domain to which the business filter for which you want to view derived metrics belongs. Click the business filter. On the View Attributes tab, move the pointer over the icon and select **Related Derived Metrics**.
4. On the **Derived Metrics** tab of the Object Explorer tab, view all the derived metrics that are created based on the business filter.

9.9.1.7.5. Modify a business filter

A business filter is used to define the scope of business to be measured. This topic describes how to modify a business filter.

Prerequisites

Business filters are created. For more information, see Create a business filter.

Context

After the computing logic of a business filter is modified, the statistical analysis logic for all the derived metrics that are created based on the business filter is also updated.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Business Filters** section.

   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.

   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.
3. Open the **Change Business Filter** dialog box by using one of the following methods:
   - In the **Business Filters** section, click the data domain to which the business filter that you want to modify belongs. Click the business filter. On the View Attributes tab, click **Change**.
   - In the **Business Filters** section, click the data domain to which the business filter that you

want to modify belongs. Move the pointer over the icon next to the business filter and select **Change**.

- In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter that you want to modify and click the icon in the **Actions** column.

4. In the **Change Business Filter** dialog box, modify the parameters as needed. For more information, see Create a business filter.
5. Submit the business filter.
   i. After you modify the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. (Optional)Publish the business filter.
   - If the current project is in Dev mode, publish the business filter to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the business filter after you submit it.

9.9.1.7.6. Unpublish and delete a business filter

This topic describes how to unpublish, unpublish and delete, and delete business filters in different states.

Prerequisites

Business filters are created. For more information, see Create a business filter.

Context

- A business filter may be in the following states:
  - After you create and save a business filter, it enters the **Draft** state.
  - After you submit a business filter, it enters the **Submitted** state.
  - After you modify and save a business filter in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a business filter in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing business filters:
  - You can unpublish only the business filters in the **Developing** or **Submitted** state.
  - You can unpublish only the business filters without derived metrics.
- Limits on deleting business filters: You can delete only the business filters in the **Draft** state.
- Limits on unpublishing and deleting business filters:
  - You can unpublish and delete only the business filters in the **Developing** or **Submitted** state.
  - You can unpublish and delete only the business filters without derived metrics.

Delete a business filter

1. Log on to the Dataphin console.

2. Go to the **Business Filters** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.You can skip this step if the current project is in **Dev** or **Basic** mode and is not the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Business Filters section appears.
3. Open the **Tip** dialog box for deleting a business filter by using one of the following methods:
   - In the **Business Filters** section, click the data domain to which the business filter that you want to delete belongs. Move the pointer over the icon next to the business filter and select **Delete**.
   - In the **Business Filters** section, click the data domain to which the business filter that you want to delete belongs. Click the business filter. On the View Attributes tab, move the pointer over the icon and select **Delete**.
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter that you want to delete and choose > **Delete** in the **Actions** column.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Unpublish a business filter

1. Open the **Tip** dialog box for unpublishing a business filter by using one of the following methods:
   - In the **Business Filters** section, click the data domain to which the business filter that you want to unpublish belongs. Move the pointer over the icon next to the business filter and select **Unpublish**.
   - In the **Business Filters** section, click the data domain to which the business filter that you want to unpublish belongs. Click the business filter. On the View Attributes tab, move the pointer over the icon and select **Unpublish**.
   - In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter that you want to unpublish and choose > **Unpublish** in the **Actions** column.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

Unpublish and delete a business filter

1. Open the **Tip** dialog box for unpublishing and deleting a business filter by using one of the

following methods:

- In the **Business Filters** section, click the data domain to which the business filter that you want to unpublish and delete belongs. Move the pointer over the icon next to the business filter and select **Unpublish and Delete**.
- In the **Business Filters** section, click **Business Filters Object List** in the lower part. The **Business Filters** tab of the **Object Explorer** tab appears. Find the business filter that you want to unpublish and delete and choose > **Unpublish and Delete** in the **Actions** column.
- In the **Business Filters** section, click the data domain to which the business filter that you want to unpublish and delete belongs. Click the business filter. On the View Attributes tab, move the pointer over the icon and select **Unpublish and Delete**.

2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.1.8. Data standardization: Derived metrics

9.9.1.8.1. Create a derived metric

Derived metrics are used to define the scope of business for atomic metrics. This topic describes how to create a derived metric.

Prerequisites

Data domains are created. For more information, see Create a data domain.

Context

Dataphin allows you to create derived metrics in a standard and unambiguous manner. Note the following information about derived metrics:

- A derived metric consists of the following elements: atomic metric, business filter, statistical period, and statistic granularity.
- Dataphin allows you to create derived metrics based on logical dimension tables or logical fact tables.
- A derived metric belongs to only one source logical table in a business unit.
- An atomic metric and a derived metric may belong to logical tables in different data domains. Therefore, a derived metric may belong to multiple data domains.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Derived Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Derived Metrics section appears.

3. Go to the **Create Derived Metric** tab by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Standardization > Derived Metrics**.
   - In the **Derived Metrics** section, click the icon next to **Derived Metrics**.
   - In the right-side workspace of the Develop page, click the icon below **Derived Metric**.
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Click **Create Derived Metric**.
4. Create a derived metric.
   i. On the **Create Derived Metric** tab, select an atomic metric.
   ii. Click **Next**.
   iii. In the **Define Derived Metric** step, set the parameters as required.

Parameter	Description
Granularity	The statistic granularity of the derived metric. You can click **Add Statistic Granularity** to add statistic granularities for the derived metric. A derived metric supports at most three statistic granularities.
Statistical Period	The statistical period of the derived metric. You can click **Add Statistical Period** to add statistical periods for the derived metric. A derived metric supports at most three statistical periods.
Business Filter	The business filter of the derived metric. You can click **Add Business Filter** to add business filters for the derived metric. A derived metric supports at most three business filters.

   iv. After you set the parameters, click **Preview Derived Metric**.
5. (Optional)In the **Change Derived Metric** section, set the **Derived Metric Name** and **Derived Metric Display Name** parameters.
6. Save and submit the derived metric.
   i. Click **Save** to save the derived metric.
   ii. Click **Submit** to submit the derived metric.
   iii. In the dialog box that appears, enter your comments.
   iv. Click **OK**.
7. (Optional)Publish the derived metric.
   - If the current project is in Dev mode, publish the derived metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the derived metric after you submit it.

9.9.1.8.2. Modify a derived metric

Derived metrics are used to define the scope of business for atomic metrics. This topic describes how to modify a derived metric.

Prerequisites

Derived metrics are created. For more information, see Create a derived metric.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Derived Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Derived Metrics section appears.
3. Open the **Change Derived Metric** dialog box by using one of the following methods:
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to modify belongs. Click the derived metric. On the View Attributes tab, click **Change**.
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to modify belongs. Move the pointer over the icon next to the derived metric that you want to modify and select **Change**.
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Find the derived metric that you want to modify and click the icon in the **Actions** column.
4. In the **Change Derived Metric** dialog box, modify the parameters as needed. For more information, see Create a business filter.
5. (Optional)Save the derived metric.If you do not want to submit the modified derived metric, click **Save**.
6. Submit the derived metric.
   i. Click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
7. (Optional)Publish the derived metric.
   - If the current project is in Dev mode, publish the derived metric to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the derived metric after you submit it.

9.9.1.8.3. Unpublish and delete a derived metric

This topic describes how to unpublish, unpublish and delete, and delete derived metrics in different states.

Prerequisites

Derived metrics are created. For more information, see Create a derived metric.

Context

- A derived metric may be in the following states:
  - After you create and save a derived metric, it enters the **Draft** state.
  - After you submit a derived metric, it enters the **Submitted** state.
  - After you modify and save a derived metric in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a derived metric in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing, deleting, and unpublishing and deleting derived metrics:
  - You can unpublish only the derived metrics in the **Developing** or **Submitted** state.
  - You can delete only the derived metrics in the **Draft** state.
  - You can unpublish and delete only the derived metrics in the **Developing** or **Submitted** state.

Delete a derived metric

1. Log on to the Dataphin console.
2. Go to the **Derived Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Derived Metrics section appears.
3. Open the **Tip** dialog box for deleting a derived metric by using one of the following methods:
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to delete belongs. Move the pointer over the icon next to the derived metric that you want to delete and select **Delete**.
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Find the derived metric that you want to delete and click the icon in the **Actions** column.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Unpublish a derived metric

1. Open the **Tip** dialog box for unpublishing a derived metric by using one of the following methods:
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to unpublish belongs. Move the pointer over the icon next to the derived metric that you want to unpublish and select **Unpublish**.
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to unpublish belongs. Click the derived metric. On the View Attributes tab, move the pointer over the icon and select **Unpublish**.
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Find the derived metric that you want to unpublish and click the icon in the **Actions** column.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

Unpublish and delete a derived metric

1. Open the **Tip** dialog box for unpublishing and deleting a derived metric by using one of the following methods:
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to unpublish and delete belongs. Move the pointer over the icon next to the derived metric that you want to unpublish and delete and select Unpublish and Delete.
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Find the derived metric that you want to unpublish and delete and click the icon in the **Actions** column.
   - In the **Derived Metrics** section, click the data domain to which the derived metric that you want to unpublish and delete belongs. Click the derived metric. On the View Attributes tab, move the pointer over the icon and select **Unpublish and Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.1.8.4. View the logical aggregate table associated with a derived metric

Dataphin allows you to view the logical aggregate table that is associated with a derived metric. This topic describes how to view the logical aggregate table that is associated with a derived metric.

Prerequisites

Derived metrics are created. For more information, see Create a derived metric.

Context

Dataphin can aggregate only the derived metrics in the **Submitted** state to logical aggregate

tables. Therefore, you can view only the logical aggregate table that is associated with a derived metric in the **Submitted** state.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Derived Metrics** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Derived Metrics section appears.
3. Go to the details page of the logical aggregate table that is associated with a derived metric by using one of the following methods:
   - In the **Derived Metrics** section, click **Derived Metrics Object List** in the lower part. The **Derived Metrics** tab of the **Object Explorer** tab appears. Find the derived metric for which you want to view the associated logical aggregate table and choose > **View Aggregate Table** in the **Actions** column.
   - In the **Derived Metrics** section, click the data domain to which the derived metric for which you want to view the associated logical aggregate table belongs. Click the derived metric. On the View Attributes tab, click **View Relative Aggregate Tables**.
4. On the tab that appears, view information about the logical aggregate table that is associated with the derived metric.

9.9.1.9. Logical tables: Logical aggregate tables

9.9.1.9.1. Create a logical aggregate table

A logical aggregate table stores statistics about a statistic granularity. This topic describes how to create a logical aggregate table.

Prerequisites

- Dimensions are created. For more information, see Create a dimension.
- Derived metrics are created. For more information, see Create a derived metric.

Context

Note the following information about logical aggregate tables:

- A logical aggregate table consists of statistical metrics at a specific statistic granularity.
- All statistical metrics in a logical aggregate table have the same statistic granularity.
- If the statistic granularity of a derived metric is the same as that of a logical aggregate table, Dataphin automatically associates the derived metric with this logical aggregate table.

The statistical metrics of a logical aggregate table are categorized into the following types:

- Derived metrics that are automatically aggregated by Dataphin
- Custom metrics

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Aggregate Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Aggregate Tables section appears.
3. Open the **Create Logical Aggregate Table** dialog box by using one of the following methods:
   - In the left-side navigation pane, click the icon next to the project name and choose **Logical Tables > Logical Aggregate Tables**.
   - In the **Logical Aggregate Tables** section, click the icon next to **Logical Aggregate Tables**.
   - In the right-side workspace of the Develop page, click the icon below **Logical Aggregate Table**.
4. In the **Create Logical Aggregate Table** dialog box, perform the following steps:
   i. In the **Primary Key Information** step, set the **Granularity** parameter.You can select one or more submitted or published dimensions as the statistic granularity.
   ii. Click **Next**.
   iii. In the **Logical Aggregate Table Description** step, enter the description of the logical aggregate table.
5. Submit the logical aggregate table.
   i. After you set the parameters in the **Create Logical Aggregate Table** dialog box, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

9.9.1.9.2. Modify a logical aggregate table

A logical aggregate table stores statistics about a statistic granularity. This topic describes how to modify a logical aggregate table.

Prerequisites

Logical aggregate tables are created. For more information, see Create a logical aggregate table.

Context

You can perform the following operations on a logical aggregate table:

- Add derived metrics.
- Add custom metrics.
- Modify derived metrics.
- View details of derived metrics and custom metrics.
- Move derived metrics or custom metrics to a specified category.
- Delete custom metrics.

Add a derived metric

1. Log on to the Dataphin console.
2. Go to the **Logical Aggregate Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Aggregate Tables section appears.
3. In the **Logical Aggregate Tables** section, move the pointer over the icon next to the logical aggregate table that you want to modify and select **Change**.
4. Steal the lock of the logical aggregate table.
   - If the logical aggregate table is locked by yourself, you do not need to steal the lock.
   - If the logical aggregate table is locked by another user, click the icon in the upper-right corner to steal the lock.
5. On the configuration tab of the logical aggregate table, click the **Derived Metrics** tab.
6. On the **Derived Metrics** tab, click **Create Derived Metric**.
7. Create, submit, and publish a derived metric. For more information, see Create a logical aggregate table.
8. Submit and publish the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.

iii. Click **OK**.

iv. Publish the logical aggregate table.

- If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

Modify a derived metric

1. On the **Derived Metrics** tab of the configuration tab of the logical aggregate table, find the derived metric that you want to modify and click Details in the **Actions** column. On the View Attributes tab, click **Change**.
2. In the **Change Derived Metric** dialog box, set the parameters and then submit and publish the derived metric. For more information, see Modify a derived metric.
3. Submit the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
4. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

View details of a derived metric

1. On the **Derived Metrics** tab of the configuration tab of the logical aggregate table, find the derived metric for which you want to view details and click **Details** in the **Actions** column.
2. On the View Attributes tab, view information about the derived metric, including the business unit and project to which it belongs and its associated atomic metrics.

Move a derived metric to a specified category

1. On the **Derived Metrics** tab of the configuration tab of the logical aggregate table, find the derived metric that you want to move and click **Category** in the **Actions** column.
2. In the **Categorize Metric** dialog box, enter a group name.
3. Click **Create** to create a group.
4. Set the Categorize To parameter to the created group and click **OK**.
5. Submit the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.

6. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

Add a custom metric

1. On the configuration tab of the logical aggregate table, click the **Custom Metrics** tab.
2. On the **Custom Metrics** tab, click **Create Custom Metric**.
3. In the **Create Custom Metric** dialog box, perform the following steps:
   i. In the **Primary Key Information** step, select the source physical table and set the primary key and association logic.
   ii. Click **Associate and Go to Next Step**.
   iii. In the **Logical Aggregate Table Description** step, select a field from the section on the left and click the > sign.
   iv. Click **OK**.
4. Submit the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
5. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

View details of a custom metric

1. On the **Custom Metrics** tab of the configuration tab of the logical aggregate table, find the custom metric for which you want to view details and click **Details** in the **Actions** column.
2. On the View Attributes tab, view information on the **Table Structure** and **Table Information** tabs.

Delete a custom metric

1. On the **Custom Metrics** tab of the configuration tab of the logical aggregate table, find the custom metric that you want to delete and click **Delete** in the **Actions** column.
2. In the message that appears, click **OK**.
3. Submit the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.

iii. Click **OK**.

4. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

Move a custom metric to a specified category

1. On the **Custom Metrics** tab of the configuration tab of the logical aggregate table, find the custom metric that you want to move and click **Category** in the **Actions** column.
2. In the **Categorize Metric** dialog box, enter a group name.
3. Click **Create** to create a group.
4. Set the Categorize To parameter to the created group and click **OK**.
5. Submit the logical aggregate table.
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to submit the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
6. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

9.9.1.9.3. Configure a logical table conversion task

To configure the custom parameters for a logical aggregate table, you can configure a logical table conversion task for the table. This topic describes how to configure a logical table conversion task for a logical aggregate table.

Prerequisites

Logical aggregate tables are created. For more information, see Create a logical aggregate table.

Context

The default retention period of a logical aggregate table is 36,000 days. You cannot modify the retention period.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Aggregate Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.

iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Aggregate Tables section appears.

3. In the **Logical Aggregate Tables** section, move the pointer over the icon next to the logical aggregate table for which you want to configure a logical table conversion task and select **Change**.
4. On the configuration tab of the logical aggregate table, click **Logical Table Conversion Task Settings** in the top navigation bar.
5. On the **Logical Table Conversion Task Settings** tab, configure the custom parameters. For more information, see Create a logical aggregate table.
6. Submit the logical aggregate table.

   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to save the logical aggregate table.

   ii. In the dialog box that appears, enter your comments.

   iii. Click **OK**.
7. (Optional)Publish the logical aggregate table.
   - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

9.9.1.9.4. View details of a logical aggregate table

A logical aggregate table stores statistics about a statistic granularity. This topic describes how to view details of a logical aggregate table.

Prerequisites

Logical aggregate tables are created. For more information, see Create a logical aggregate table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Logical Aggregate Tables** section.

   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.

   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.

iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Aggregate Tables section appears.

3. In the **Logical Aggregate Tables** section, move the pointer over the icon next to the logical aggregate table for which you want to view details and select **Change**.
4. On the configuration tab of the logical aggregate table, click **Table Information** in the top navigation bar.In the **Logical Aggregate Table Details** pane, you can perform the following operations:
   - View the primary key in the **Primary Key Information** section.
   - View the name, display name, and description in the **Logical Aggregate Table Description** section.
   - Click the dimension name next to **Granularity** to view the dimension information.

   Click **OK**. You can modify the description of the logical aggregate table in the **Logical Aggregate Table Description** section.
5. (Optional)Submit and publish the logical aggregate table.If you modify the description of the logical aggregate table in the **Logical Aggregate Table Description** section, submit and publish the logical aggregate table. To submit and publish a logical aggregate table, perform the following steps:
   i. On the configuration tab of the logical aggregate table, click the icon in the upper-right corner to save the logical aggregate table.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
   iv. Publish the logical aggregate table.
      - If the current project is in Dev mode, publish the logical aggregate table to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the logical aggregate table after you submit it.

9.9.1.9.5. Delete a logical aggregate table

A logical aggregate table stores statistics about a statistic granularity. This topic describes how to delete a logical aggregate table.

Prerequisites

Logical aggregate tables are created. For more information, see Create a logical aggregate table.

Context

You can delete only the logical aggregate tables that have no derived metrics or custom metrics.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Logical Aggregate Tables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. Do not select the **Data_distill** project.
   iii. On the **Develop** page, click the **Standards and Modeling** tab in the left-side navigation pane.
   iv. On the **Standards and Modeling** tab, move the pointer over the left-side navigation submenu and click the icon. The Logical Aggregate Tables section appears.
3. In the Logical Aggregate Tables section, move the pointer over the icon next to the logical aggregate table that you want to delete and click **Delete**.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

9.9.1.10. Standards and modeling: Modeling engine

Based on the built-in rules of the modeling engine, Dataphin verifies the logical tables during logical table creation, submission, and logical tables task scheduling, and then converts these logical tables into physical tables. Therefore, you can query these physical tables. Modeling engine is used to ensure the model quality and control of the R&D process. Standard, reliable, stable, and efficient models rely on the intelligent modeling engine service.

Query logical tables using SQL statements

Dataphin supports the query of a logical table based on the snowflake schema. The snowflake schema can query data based on [logical model. dimension-associated field. dimension-associated field.....attribute] (such as order.buyer.membership type.type). This can improve the coding efficiency of the SQL queries.

```
SELECT a.bob_cw_bob_ooo,
  a.ds_1d__t_dts,
  a.ds_cw_234444444556,
  a.dim_cc.code
FROM
  LD_retail.dws_cc a
where a.dim_cc.value is NOT NULL
```

- **Query logical fact tables or logical dimension tables**

- SQL syntax:

```
SELECT Column,
    LTable.Column,  // Logical table name.field name
    LTable.Role(relation - dimension).....Dimension_Column,  //Logical table name.role name (dim
ension-associated field name).dimension attribute name
FROM  LD.LTable  //Business unit name.logical table name
WHERE  ...
```

```
SELECT Column,
    LTable_Alias.Column,
    LTable_Alias.Role(relation - dimension).....Dimension_Column
FROM  LD.LTable  Alias
WHERE  ...
```

- SQL format:

```
SELECT a. dim_industry.in_name,
    a.dim_industry.IN_ID,
    name,
    ceo_name,
    company_id
FROM  LD_demo.dim_company.dim_industry.in_name  a
WHERE  ds = '${bizdate}'
LIMIT  100;
```

- **Query logical aggregate tables**
  - SQL syntax:

```
SELECT Column,
    LTable.Column,
    LTable.Dimension(granularity).....Dimension_Column,
FROM  LD.LTable
WHERE  ...
```

```
SELECT Column,
    Alias.Column,
    Alias.Dimension (granularity).....Dimension_Column
FROM  LD.LTable Alias
WHERE  ...
```

- SQL format:

```
SELECT customer_id,
      c.dim_customer.address_line1,
      c.dim_customer.dim_tax_rate.tx_name,
      crt_trd_comm_amt_30d_trd_valid,
      acct_cnt_td_actv_account,
      watch_sec_cnt_td_actv_watches,
      avg_crt_trd_comm_7d,
      crt_trd_acct_cnt_7d,
      crt_trd_cnt_1d,
      avg_crt_trd_comm_1d,
      crt_trd_comm_amt_1d_trd_valid,
      crt_trd_acct_cnt_1d,
      watch_sec_cnt_td_inac_watches,
      watch_sec_cnt_td,
      acct_cnt_td,
      crt_trd_cnt_7d
FROM  LD_DEMO.DWS_CUSTOMER c
WHERE ds='${bizdate}'
LIMIT 100;
```

Note

- If two dimension-associated fields referenced by dimension roles in a logical dimension table have the same name, you need to set different display names. This is to prevent conflicts with field names in an SQL statement.
- Only logical dimension tables of parent dimensions can be queried. If a dimension is a child dimension, you can only use the child dimension based on the logical dimension tables of the parent dimension, or dimension-associated fields.
- Fields contained in the logical dimension table of a child dimension are displayed in the list of logical dimension table fields of the parent dimension.
- A dimension that composes the statistic granularity of a logical aggregate table can be referenced in a similar way to associate a dimension to a logical aggregate table. You can obtain the [dimension _ primary key] data by running the *SELECT* *command. To prevent slow query of a full table, we recommend that you obtain the primary key by using the *SELECT logical table name.dimension table name.primary key* statement and obtain the attribute information by using the *SELECT logical table name.dimension table name.attribute* statement.

Verify the repetition of the computing logic

One of the core benefits of Dataphin is the unique definition, which requires that the name and computing logic are unique. When you submit a standard definition or logical tables, Dataphin will verify the name, display name, and computing logic of the object. You can only submit them when the definition or objects are unique. If duplicate computing logic exists, the system displays a message to help you avoid creating objects with the same name but different meanings or vice versa.

When you submit and publish objects, the system parses the requests based on the abstract syntax tree (AST) and verifies the repetition of computing logic (or expressions). The objects include time periods, dimensions, logical dimension table fields, logical fact tables, logical fact table fields, atomic metrics, business filters, and custom metrics of logical aggregate tables. If duplicate computing logic exists, the system displays a message to indicate that the object computing logic is repeated.

Refresh model versions in a dynamic way

Dynamic refresh can improve the efficiency of submitting and converting logical tables. It can help increase the flexibility of modifying logical tables, and reduce the computing engine resource consumption for physical table changes, historical data changes, and migration. During SQL task scheduling, the system automatically identifies and routes to the corresponding physical tables based on the latest logical table conversion to obtain the required data.

To view the version information of a logical table, you can select **Versions** on the top of the logical table tab.

9.9.2. Batch processing

9.9.2.1. Overview

Batch processing overview

Dataphin allows you to create and run batch processing tasks during the data development process and manage the functions and files that are used in these tasks.

Dataphin allows you to create batch processing tasks of the **MAX_COMPUTRE_SQL**, **MAX_COMPUTE_MR**, **SPARK_JAR_ON_MAX_COMPUTE**, **SHELL**, **PYTHON**, and **VIRTUAL** types. You can synchronize data, create destination tables, create complex data models, and process logical tables by using batch processing tasks.

The development process of a batch processing task consists of the following steps:

1. Create resources. For more information, see Create a resource.
2. Create batch processing functions. For more information, see Create a UDF.
3. Create batch processing tasks. For more information, see Create a batch processing task.

9.9.2.2. Resource management

9.9.2.2.1. Create a folder

This topic describes how to create a folder for storing resources.

Context

Folders help you categorize and manage resources.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, click the icon next to **Resource Management**.
7. In the **Create Folder** dialog box, enter a folder name and select a directory.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.2.2.2. Create a resource

Dataphin provides the Resource Management section on the Develop page for you to store and manage files that are required during coding, such as JAR, JSON, and Python files. This topic describes how to create a resource.

Context

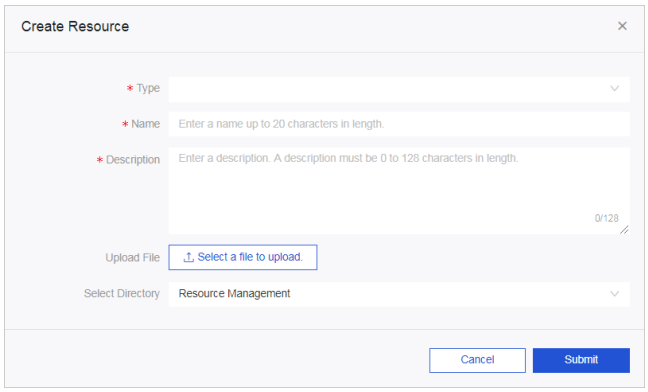
To reference files based on your business needs when you develop batch processing tasks, you can upload these files to Dataphin.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

   > **Note** If the current project is in Dev mode, publish the resource to the corresponding project in Prod mode. Then, you can view published resources in the production environment.

4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. Open the **Create Resource** dialog box by using one of the following methods:
   - In the Resource Management section, click the icon next to **Resource Management**.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing** > **Resources**.
   - In the right-side workspace of the **Develop** page, click the icon next to **Resources**.
7. In the **Create Resource** dialog box, set the parameters as required.



Parameter	Description

Parameter	Description
Type	The type of the file to upload. Valid values: file, jar, Python, and others. The maximum size of each file is 50 MB. You can upload up to 1,000 files. Note If the network connection is unstable, we recommend that you upload files that are not larger than 100 MB.
Name	The name of the resource to create. The name of the resource must be unique in the project.
Description	The description of the resource.
Upload File	The file to upload. Select the file to upload based on the value of the Type parameter.
Resource Usage	The usage of the resource. Valid values: Non-UDF, Batch Processing UDF, and Stream Processing UDF.
Select Directory	The directory for storing the resource. Select an existing directory.

8. Submit the resource.
   i. Click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
9. (Optional)Publish the resource.
   - If the current project is in **Dev** mode, publish the resource to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in **Basic** mode, you do not need to publish the resource after you submit it.

9.9.2.2.3. Modify a resource

To modify an uploaded file, you can modify the corresponding resource and upload the modified file to replace the original file. This topic describes how to modify a resource.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.

5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to modify and select **Change**.
7. In the **Change Resource** dialog box, modify the parameters as needed. You can change the value of the **Description** parameter and re-upload a file.
8. Submit the resource.
   i. Click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
9. (Optional)Publish the resource.
   - If the current project is in **Dev** mode, publish the resource to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in **Basic** mode, you do not need to publish the resource after you submit it.

9.9.2.2.4. Move a resource

This topic describes how to move a resource to another directory.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination directory.
8. Click **OK**.

9.9.2.2.5. Delete a resource

You can delete a resource that is no longer used. This topic describes how to delete a resource.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to delete and select **Delete**.

   Note You cannot delete the resource if it is referenced by a task.
7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

9.9.2.3. Manage batch processing functions

9.9.2.3.1. Create a folder

This topic describes how to create a folder for storing functions.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Functions** section appears.
6. In the **Batch Processing Functions** section, click the icon next to **Batch Processing Functions**.
7. In the **Create Folder** dialog box, enter a folder name and select a directory.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Batch Processing Functions section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Batch Processing Functions section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Batch Processing Functions section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.2.3.2. Create a UDF

Dataphin provides the Batch Processing Functions section on the Develop page. In this section, you can manage SQL functions to be used in batch processing tasks, including system built-in functions that are commonly used by computing engines and user-defined functions (UDFs). System built-in functions cannot be modified. This topic describes how to create a UDF.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Functions** section appears.
6. Open the **Create Function** dialog box by using one of the following methods:
   - In the Batch Processing Functions section, click the icon next to **Batch Processing Functions**.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing > Batch Processing Functions**.
   - In the right-side workspace of the **Develop** page, click the icon below **Batch Processing Function**.
7. In the **Create Function** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the function to create. The name can contain letters, digits, and underscores (_), and must start with a letter.
Resource	The resource to be referenced by the function. You can view the resources that are available for the current project in the drop-down list. Note - You can create only functions that reference Python or JAR files. - If you select multiple resources, they must be of the same type. - If you do not have any resources, you must create one. For more information, see Create a resource.
Class	The class of the function to create. You can extract the class from the resources in the MaxCompute computing engine. Example: `test_udf.UDFGETSrcId`.
Type	The type of the function to create. Valid values: **Window**, **Statistic**, **Numeric**, **String**, **DateTime**, **ip address-related function**, **URL**, **Codec**, **Business**, and **Other**.
Function Syntax	The syntax of the function to create. The syntax of a function is the format for referencing the function. Example: `bigintweekday (datetime date)`.

Parameter	Description
Description	The description of the function to create. select get_week_date("20170810",0,2), -- Query the date of the Tuesday in t he week of August 10, 2017. from cndata.dual
Select Directory	The directory of the function to create. By default, the User-Defined Function directory is selected. You can also select another directory from the drop-down list.

8. Submit the UDF.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**. The UDF is created and registered with MaxCompute.

   Note If the resources referenced by UDFs are updated, submit the UDFs again so that the UDFs registered with MaxCompute are updated.

   You can reference the UDF in an SQL statement and execute the statement in a task on the **Ad Hoc Query** tab to check whether the function meets your expectation. For more information, see Create an ad hoc query task. The following SQL statement is used as an example:

```
select
get_week_date("20170810",0,2), -- Query the date of the Tuesday in the week of August 10, 2017.
from cndata.dual
```

9. (Optional)Publish the UDF.
   - If the current project is in **Dev** mode, publish the UDF to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in **Basic** mode, you do not need to publish the UDF after you submit it.

9.9.2.3.3. Modify a UDF

Dataphin provides the Batch Processing Functions section on the Develop page. In this section, you can manage SQL functions to be used in batch processing tasks, including system built-in functions that are commonly used by computing engines and UDFs. System built-in functions cannot be modified. This topic describes how to modify a UDF.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Functions** section appears.
6. In the **Batch Processing Functions** section, move the pointer over the icon next to the UDF that you want to modify and select **Change**.
7. In the **Change Function** dialog box, modify the parameters as needed. You can change the values of the **Resource**, **Class**, **Type**, **Function Syntax**, and **Description** parameters. For more information, see Create a UDF.
8. Submit the UDF.
   i. After you modify the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
9. (Optional)Publish the UDF.
   - If the current project is in Dev mode, publish the UDF to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the UDF. After you submit a UDF, it can be referenced in batch processing tasks that are scheduled.

9.9.2.3.4. Move a UDF

This topic describes how to move a UDF to another directory.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Functions** section appears.

6. In the **Batch Processing Functions** section, move the pointer over the icon next to the UDF that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination directory.
8. Click **OK**.

9.9.2.3.5. Delete a UDF

Dataphin provides the Batch Processing Functions section on the Develop page. In this section, you can manage SQL functions to be used in batch processing tasks, including system built-in functions that are commonly used by computing engines and UDFs. System built-in functions cannot be modified. This topic describes how to delete a UDF.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Functions** section appears.
6. In the **Batch Processing Functions** section, move the pointer over the icon next to the UDF that you want to delete and select **Delete**.

   > **Note** You cannot delete a UDF that is referenced in a task.

7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

9.9.2.4. Create a folder for storing batch processing tasks

This topic describes how to create a folder for storing batch processing tasks.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
4. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Tasks** section appears.

5. In the Batch Processing Tasks section, click the icon next to **Batch Processing Tasks**.
6. In the **Create Folder** dialog box, enter a folder name and select a directory.
7. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Batch Processing Tasks section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Batch Processing Tasks section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Batch Processing Tasks section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.2.5. Create batch processing tasks

9.9.2.5.1. Create a batch processing task of the MAX_COMPUTE_SQL type

This topic describes how to create a batch processing task of the MAX_COMPUTE_SQL type in Dataphin.

Context

You can create a batch processing task of the MAX_COMPUTE_SQL type based on your needs to process existing data and obtain required data.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Tasks** section appears.
3. In the Batch Processing Tasks section, click the icon next to **Batch Processing Tasks** and select **MAX_COMPUTE_SQL**. The Create Item dialog box appears.
4. Set the task parameters, write the code of the task, and then run the code.
   i. In the **Create Item** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the batch processing task, for example, SQL.
Schedule Type	The scheduling type of the task. Valid values: **Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   ii. Click **OK**.
   iii. On the **Code Editor** tab, write the SQL code of the task.
   iv. Click **Run** in the upper-right corner to run the code.
5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the task.
   i. Click the icon in the upper-right corner of the Code Editor tab to save the task.
   ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.
7. (Optional)Publish the task.
   - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the task. The task can be

scheduled in the production environment after you submit it.

9.9.2.5.2. Create a batch processing task of the MAX_COMPUTE_MR type

This topic describes how to create a batch processing task of the MAX_COMPUTE_MR type in Dataphin.

Prerequisites

JAR packages are uploaded. For more information, see Create a resource.

Context

A batch processing task of the MAX_COMPUTE_MR type must reference JAR packages. Before you create a batch processing task of the MAX_COMPUTE_MR type, you must upload the JAR packages to be referenced by the task.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing Tab, move the pointer over the left-side navigation submenu and click the icon. The **Batch Processing Tasks** section appears.
3. In the Batch Processing Tasks section, click the icon next to **Batch Processing Tasks** and select **MAX_COMPUTE_MR**. The Create Item dialog box appears.
4. Set the task parameters, write the code of the task, and then run the code.
   i. In the **Create Item** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the batch processing task, for example, MR.
Schedule Type	The scheduling type of the task.**Valid values: Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   ii. Click **OK**.

iii. On the **Code Editor** tab, write the code of the task.In this example, write the following code:

```
@resource_reference{"mr_odps.jar"}
add jarmr_odps.jar as momo.jar -f;
jar -resources  momo.jar -classpath mr_odps.jar hive.WordCountOdps wc_in wc_out;
```

iv. Click **Run** in the upper-right corner to run the code.

5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the task.
   i. Click the icon in the upper-right corner of the Code Editor tab to save the task.
   ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.
7. (Optional)Publish the task.
   - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.9.2.5.3. Create a batch processing task of the SPARK_JAR_ON_MAX_COMPUTE type

This topic describes how to create a batch processing task of the SPARK_JAR_ON_MAX_COMPUTE type in Dataphin.

Prerequisites

JAR or Python packages are uploaded. For more information, see Create a resource.

Context

A batch processing task of the SPARK_JAR_ON_MAX_COMPUTE type must reference JAR or Python packages. Before you create a batch processing task of the SPARK_JAR_ON_MAX_COMPUTE type, you must upload the JAR or Python packages to be referenced by the task.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.

iv. On the left-side navigation submenu, click the icon.

3. In the **Batch Processing Tasks** section, click the icon next to Batch Processing Tasks and select **SPARK_JAR_ON_MAX_COMPUTE**.
4. Set the task parameters, write the code of the task, and then run the code.
   i. In the **Create Item** dialog box, set the task parameters.

Parameter	Description
Name	The name of the batch processing task, for example, Spark_python.
Schedule Type	The scheduling type of the task. Valid values: **Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   ii. Click **OK**.

   iii. On the **Code Editor** tab, write the code of the task.In this example, write the following code:

```
@resource_reference{"spark.py"}
spark-submit
--deploy-mode cluster
--conf spark.hadoop.odps.task.major.version=cupid_v2
--conf spark.hadoop.odps.end.point=http://service.cn.maxcompute.aliyun.com/api
--conf spark.hadoop.odps.runtime.end.point=http://service.cn.maxcompute.aliyun-inc.com/api
--master yarn
spark.py
```

   In the sample code, `required_resource{}` specifies the JAR or Python package to be referenced, and all the other statements are fixed.

   iv. Click **Run** in the upper-right corner to run the code.

5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the task.

i. Click the icon in the upper-right corner of the Code Editor tab to save the task.

ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.

7. (Optional)Publish the task.
   - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.9.2.5.4. Create a batch processing task of the SHELL type

This topic describes how to create a batch processing task of the SHELL type in Dataphin.

Prerequisites

JSON packages are uploaded. For more information, see Create a resource.

Context

A batch processing task of the SHELL type must reference JSON packages. Before you create a batch processing task of the SHELL type, you must upload the JSON packages to be referenced by the task.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the left-side navigation submenu, click the icon.
3. In the **Batch Processing Tasks** section, click the icon next to Batch Processing Tasks and select **SHELL**.
4. Set the task parameters, write the code of the task, and then run the code.

i. In the **Create Item** dialog box, set the task parameters.

Parameter	Description
Name	The name of the batch processing task, for example, DataX.
Schedule Type	The scheduling type of the task. In this example, select **Recurring Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

ii. Click **OK**.

iii. On the **Code Editor** tab, write the code of the task.

In this example, write the following code:

```
@required_resource{required_memory=2Gb;required_cpus=1.0}
@resource_reference{"Datax.json"}
python $DATA_HOME/bin/datax.py --jvm '-Xms2g -Xmx2g' Datax.json
```

In the sample code:

- required_resource{} specifies the computing resources to be allocated to the task.
- $DATAX_HOME is a system built-in parameter that specifies the installation directory of the sync task. The entry class of the sync task is in the bin subdirectory of the installation directory.
- `--jvm '-Xms2g -Xmx2g'` specifies the memory size of the Java virtual machine (JVM) when a sync task is running. We recommend that you set the memory size the same as the value of required_memory in required_resource.

Enter the following code for a sync task that requires a small amount of computing resources:

```
@resource_reference{"Datax.json"}
python $DATA_HOME/bin/datax.py Datax.json
```

iv. Click **Run** in the upper-right corner to run the code.

5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the task.
   i. Click the icon in the upper-right corner of the Code Editor tab to save the task.
   ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.
7. (Optional)Publish the task.

- If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.9.2.5.5. Create a batch processing task of the PYTHON type

This topic describes how to create a batch processing task of the PYTHON type in Dataphin.

Prerequisites

Python packages are uploaded. For more information, see Create a resource.

Context

A batch processing task of the PYTHON type must reference Python packages. Before you create a batch processing task of the PYTHON type, you must upload the Python packages to be referenced by the task.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the left-side navigation submenu, click the icon.
3. In the **Batch Processing Tasks** section, click the icon next to Batch Processing Tasks and select **PYTHON**.
4. Set the task parameters, write the code of the task, and then run the code.
   i. In the **Create Item** dialog box, set the task parameters.

Parameter	Description
Name	The name of the batch processing task, for example, Python.
Schedule Type	The scheduling type of the task. Valid values: **Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   ii. Click **OK**.

iii. On the Code Editor tab, write the code of the task.

In this example, write the following code:

```
#! /usr/bin/python
# -*- coding: latin-1 -*-
import time
import datetime
import base64
import hashlib
import json
import sys
import csv
from odps import ODPS
def main():
    print "Hello World"
```

We recommend that you comment out the first two lines of the Python code. This helps avoid code execution errors when the system encoding format is used.

iv. Click **Run** in the upper-right corner to run the code.

5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the task.
   i. Click the icon in the upper-right corner of the Code Editor tab to save the task.
   ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.
7. (Optional)Publish the task.
   - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.9.2.5.6. Create a zero load node of the VIRTUAL type

When you configure the scheduling policy for a node, you can configure the node to depend on the zero load node as required. This topic describes how to create a zero load node of the VIRTUAL type in Dataphin.

Context

Before you build a data model, you can create a zero load node for the data model. After that, when you configure the scheduling policy for a node in the data model, you can configure the node to depend on the zero load node.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Batch Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the left-side navigation submenu, click the icon.
3. In the **Batch Processing Tasks** section, click the icon next to Batch Processing Tasks and select **VIRTUAL**.
4. Set the task parameters, write the code of the task, and then run the code.
   i. In the **Create Item** dialog box, set the task parameters.

Parameter	Description
Name	The name of the batch processing task, for example, Virtual.
Schedule Type	The scheduling type of the task. Valid values: **Recurring Node** and **One-Time Node**.
Description	The description of the task.
Select Directory	The folder for storing the task.

   ii. Click **OK**.
   iii. On the **Code Editor** tab, write the SQL code of the task.
   iv. Click **Run** in the upper-right corner to run the code.

> **Note** Dataphin does not run the code of a zero load node. Instead, Dataphin directly sets the code execution result to successful.

5. (Optional)Configure the scheduling policy.
   - If you set the Schedule Type parameter to **Recurring Node**, you must configure the scheduling policy. For more information, see Configure a scheduling policy.
     - If you set the Schedule Type parameter to **One-Time Node**, you do not need to configure the scheduling policy.
6. Save and submit the zero load node.
   i. Click the icon in the upper-right corner of the Code Editor tab to save the task.

ii. Click the icon in the upper-right corner of the Code Editor tab to submit the task.

7. (Optional)Publish the zero load node.
   - If the current project is in Dev mode, publish the node to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the node. The node can be scheduled in the production environment after you submit it.

9.9.2.6. Configure a scheduling policy

Dataphin allows you to configure the scheduling rules and dependencies of nodes to make sure that tasks can be properly scheduled. This topic describes how to configure a scheduling policy for a batch processing task.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Context

- Dataphin allows you to configure scheduling policies only for recurring tasks.
- A dependency is a semantic connection between two or more nodes. The status of an upstream node affects the running of its downstream nodes.
- After you configure dependencies for an upstream node and its downstream nodes, the downstream nodes can be run only after the upstream node is run. Before the system runs a node, the system checks the scheduling time that is configured for the node and determines whether to run the node.
- If a scheduling configuration is submitted before the specified scheduling time, the configuration takes effect after the specified scheduling time. If a dependency is configured after the specified scheduling time, the dependency takes effect for the instances that are generated the next day.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. Go to the **Scheduling Configuration** tab of the batch processing task for which you want to configure a scheduling policy.
   i. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   ii. On the left-side navigation submenu, click .
   iii. In the **Batch Processing Tasks** section, click the batch processing task for which you want to configure a scheduling policy. The Code Editor tab of the batch processing task appears.

iv. (Optional)Steal the lock of the batch processing task.

- If the batch processing task is locked by yourself, you do not need to steal the lock.
- If the batch processing task is locked by another user, click the icon in the upper-right corner to steal the lock.

v. Click the **Scheduling Configuration** tab.

5. In the **Scheduling Configuration** pane, set the parameters as required.

i. Set the parameters in the **Basic Information** section.Dataphin automatically generates the node name, node ID, node type, and owner. You cannot modify these parameters.

Parameter	Description
Description	The description of the scheduling policy.
Priority	The priority based on which the batch processing task is scheduled. Valid values: ■ **Lowest Priority** ■ **Low Priority** ■ **Medium Priority** ■ **High Priority** ■ **Highest Priority**
Parameters	The specified values for the parameters that are used in the code of the task. Dataphin allows you to specify the parameters that are used in the code of a task. The specified values are used when the task is run. You can click **Parameters and Descriptions** to know how to configure the parameters.

ii. Set the parameters in the **Scheduling Configuration** section.

Parameter	Description
Schedule Mode	The scheduling mode of the task. Valid values: ■ **Normal**: runs the task based on the specified recurrence. By default, this option is selected. ■ **Dry-run**: runs the task based on the specified recurrence. However, the scheduling system does not actually run the task but directly returns a success response. ■ **Pause Scheduling**: runs the task based on the specified recurrence. However, the scheduling system does not actually run the task but directly returns a failure response. You can select this check box if you need to suspend a task and run it later.

Parameter	Description
Recurrence	The recurrence of the task. Valid values: ■ **Day**: automatically runs the task once per day. When you create a recurring task, the task is set to run at 00:00 every day by default. You can also click the icon to specify a time for the task to be run as needed. ■ **Week**: automatically runs the task at a specified time point on specified days of each week. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not actually run the instance or consume resources but directly returns a success response. Assume that you set Recurrence to Week and specify that the task is run every Monday and Tuesday. The scheduling system generates and runs instances every Monday and Tuesday. Every Wednesday, Thursday, Friday, Saturday, and Sunday, the scheduling system generates instances and returns success responses without running the instances. ■ **Month**: automatically runs the task at a specified time point on specified days of each month. On the other days, the scheduling system still generates an instance every day to ensure the proper running of downstream instances. However, the system does not actually run the instance or consume resources but directly returns a success response. Assume that you set Recurrence to Month and specify that the task is run on the seventh day of each month. The scheduling system generates and runs an instance on the seventh day of each month. On the other days, the scheduling system generates instances and returns success responses without running the instances. ■ **Hour**: automatically runs the task at a specified interval during a specified time period or at specified time points every day. The scheduling system automatically generates instances for the task and runs the instances at the specified interval or time points. Assume that you set Recurrence to Hour, select Time Period, and set the Start, End, and Interval parameters to 00:00, 23:59, and 1, respectively. The scheduling system automatically generates instances for the task and runs an instance every hour. ■ **Minute**: automatically runs the task at a specified interval during a specified time period every day. Assume that you set Recurrence to Minute and set the Start, End, and Interval parameters to 00:00, 23:59, and 05, respectively. The scheduling system automatically generates instances for the task and runs an instance every 5 minutes.

Parameter	Description
Depend on Previous Instance	Specifies whether the current task is run after the previous instance of another task or of the current task is run. If you select Depend on Previous Instance, you must further select Current Task or Select Task. ■ If you select **Current Task**, the current task is run after the previous instance of the current task is run. ■ If you select **Select Task**, enter a keyword in the search box that appears to search for and select one or more tasks to depend on.

iii. Set the parameters in the **Dependency** section.

Parameter	Description
Start Parsing	**Start Parsing** is available for SQL tasks. After you click Start Parsing, Dataphin parses the table referenced in the code and finds the output name that is the same as the table name. Dataphin automatically configures the node corresponding to this output name as the upstream node on which the current node depends. If the project referenced in the code is a variable or no project is specified in the code, Dataphin parses the name of the production project by default to ensure stable scheduling. Assume that the name of the production project is `onedata_dev`. ■ If the code contains `select * from s_order`, the `onedata.s_order` node is parsed as the upstream node. ■ If the code contains `select * from ${onedata}.s_order`, the `onedata.s_order` node is parsed as the upstream node. ■ If the code contains `select * from onedata.s_order`, the `onedata.s_order` node is parsed as the upstream node. ■ If the code contains `select * from onedata_dev.s_order`, the `onedata_dev.s_order` node is parsed as the upstream node.

Parameter	Description
Upstream Dependency	The upstream nodes on which the current node depends. To specify an upstream node, perform the following steps: a. Click **Create Upstream Dependency**. b. In the **Create Upstream Dependency** dialog box, search for an upstream node in one of the following ways: ■ Enter a keyword to search for an upstream node whose output name contains the keyword. ■ Enter **virtual** to search for the zero load node. A root node is generated for each tenant or enterprise during initialization. The root node is a zero load node and its name starts with virtual. **Note** The output name of a node is globally unique and is not case-sensitive. c. Click **OK**. To remove a node from the upstream node list, click the icon in the **Actions** column.
Current Node	The output name for the current node. You can set multiple output names for a node, which can be used to configure dependencies for other nodes. To set an output name, perform the following steps: a. Click **Add**. b. In the **Add Output Task Nodes for Current Task Node** dialog box, enter an output name. Observe uniform rules when you set each output name for the current node. In general, set an output name in the format of `Project name.Table name`, which is not case-sensitive. This helps other users find this node when they configure the upstream dependency for their nodes. For example, if the name of the development project is `onedata_dev`, we recommend that you set the output name to `onedata.s_order`. If you set the output name to `onedata_dev.s_order`, the current node can be parsed as an upstream node for other nodes only when the code contains `select * from onedata_dev.s_order`. c. Click **OK**. You can also perform the following operations on an existing output name: ■ To delete an output name, click the icon in the **Actions** column. ■ To view the downstream nodes of the current node, click the icon in the **Actions** column.

6. Click **OK**.

9.9.2.7. Modify a batch processing task

Batch processing tasks are used to process data offline during data development. This topic describes how to modify a batch processing task.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Context

If the batch processing task that you want to modify is locked by another user, click the icon in the upper-right corner to steal the lock.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click .
6. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to modify and select **Change**.
7. Modify the batch processing task.
   i. (Optional)Steal the lock of the batch processing task.
      - If the batch processing task is locked by yourself, you do not need to steal the lock.
      - If the batch processing task is locked by another user, click the icon in the upper-right corner to steal the lock.
   ii. Modify the code or scheduling policy of the batch processing task.
8. Save and submit the batch processing task.
   i. Click the icon in the upper-right corner to save the batch processing task.
   ii. Click the icon in the upper-right corner to submit the batch processing task.
   iii. Publish the batch processing task.
      - If the current project is in Dev mode, publish the task to the corresponding project in Prod mode. For more information, see Publishing management.
      - If the current project is in Basic mode, you do not need to publish the task. The task can be scheduled in the production environment after you submit it.

9.9.2.8. Unpublish and delete a batch processing task

Batch processing tasks are used to process data offline during data development. This topic describes how to unpublish, unpublish and delete, and delete batch processing tasks in different states.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Context

- A batch processing task may be in the following states:
  - After you create and save a batch processing task, it enters the **Draft** state.
  - After you submit a batch processing task, it enters the **Submitted** state.
  - After you modify and save a batch processing task in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a batch processing task in the **Submitted** state, it enters the **Draft** state.
- Limits on unpublishing, deleting, and unpublishing and deleting batch processing tasks:
  - You can unpublish batch processing tasks only when they are in the **Developing** or **Submitted** state.
  - You can delete batch processing tasks only when they are in the **Draft** state.
  - You can unpublish and delete batch processing tasks only when they are in the **Developing** or **Submitted** state.

Unpublish a batch processing task

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to unpublish and select **Unpublish**.
7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

Unpublish and delete a batch processing task

1. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to unpublish and delete and select **Unpublish and Delete**.
2. In the **Tip** dialog box, enter your comments.

3. Click **OK**.

Delete a batch processing task

1. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.2.9. View the version information about a batch processing task

This topic describes how to view the version information about a batch processing task, including the details and code of each version.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Batch Processing Tasks** section, click the batch processing task whose version information you want to view.
7. On the configuration tab of the task, click **Versions** in the top navigation bar.
8. In the **Versions** pane, view the historical versions of the batch processing task.
   - Click **Details** in the **Actions** column of a version to view the details about the version.
   - Click **Compare Code Version** in the **Actions** column of a version to compare the code of the version with that of other versions.
   - Click **View Code** in the **Actions** column of a version to view the code of the version.

9.9.2.10. Move a batch processing task

Dataphin allows you to move a batch processing task to a specified folder. This topic describes how to move a batch processing task.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
8. Click **OK**.

9.9.2.11. Rename a batch processing task

Dataphin allows you to rename a batch processing task. This topic describes how to rename a batch processing task.

Prerequisites

Batch processing tasks are created. For more information, see Create a batch processing task.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Batch Processing Tasks** section, move the pointer over the icon next to the batch processing task that you want to rename and select **Rename**.
7. In the field that appears, enter a new name.
8. Press the Enter key.

9.9.3. Stream processing

9.9.3.1. Overview

Data Processing allows you to run and manage stream processing tasks that are created during the data development process. You can also manage the files, metatables, and templates that are used in these tasks.

Dataphin allows you to create stream processing tasks of the **FLINK_SQL** and **FLINK_TEMPLATE_SQL** types. For more information, see Create a stream processing task of the FLINK_SQL type and Create a stream processing task of the FLINK_TEMPLATE_SQL type.

To develop a stream processing task, perform the following steps:

1. Create a resource. For more information, see Create a resource.
2. Create a stream processing function. For more information, see Create a UDF.
3. Create a stream metatable. For more information, see Create a stream metatable.
4. Optional. Create a stream processing template. For more information, see Create a stream processing template.
5. Create a stream processing task. For more information, see Create a stream processing task of the FLINK_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.

> **Note** Stream processing templates can be used only by stream processing tasks of the FLINK_TEMPLATE_SQL type.

9.9.3.2. Resource management

9.9.3.2.1. Create a folder

This topic describes how to create a folder for storing resources.

Context

Folders help you categorize and manage resources.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, click the icon next to **Resource Management**.
7. In the **Create Folder** dialog box, enter a folder name and select a directory.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination directory. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Resource Management section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.3.2.2. Create a resource

Dataphin provides the Resource Management section on the Develop page for you to store and manage files that are required during coding, such as JAR, JSON, and Python files. This topic describes how to create a resource.

Context

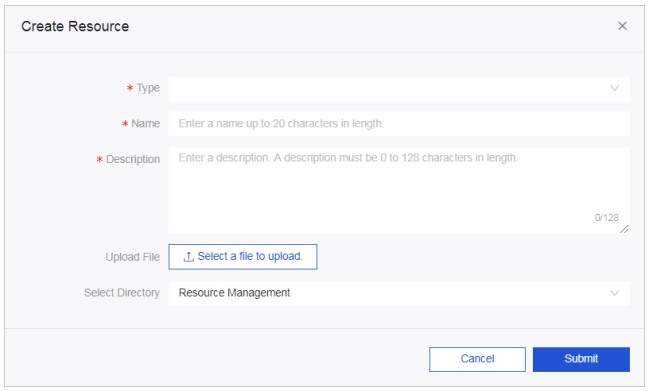
To reference files based on your business needs when you develop batch processing tasks, you can upload these files to Dataphin.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

> **Note** If the current project is in Dev mode, publish the resource to the corresponding project in Prod mode. Then, you can view published resources in the production environment.

4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the **Data Processing** tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. Open the **Create Resource** dialog box by using one of the following methods:
   - In the Resource Management section, click the icon next to **Resource Management**.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing > Resources**.
   - In the right-side workspace of the **Develop** page, click the icon next to **Resources**.
7. In the **Create Resource** dialog box, set the parameters as required.



Parameter	Description
Type	The type of the file to upload. Valid values: **file**, **jar**, **Python**, and **others**. The maximum size of each file is 50 MB. You can upload up to 1,000 files. **Note** If the network connection is unstable, we recommend that you upload files that are not larger than 100 MB.
Name	The name of the resource to create. The name of the resource must be unique in the project.
Description	The description of the resource.
Upload File	The file to upload. Select the file to upload based on the value of the **Type** parameter.
Resource Usage	The usage of the resource. Valid values: **Non-UDF**, **Batch Processing UDF**, and **Stream Processing UDF**.
Select Directory	The directory for storing the resource. Select an existing directory.

8. Submit the resource.
   i. Click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
9. (Optional)Publish the resource.
   - If the current project is in **Dev** mode, publish the resource to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in **Basic** mode, you do not need to publish the resource after you submit it.

9.9.3.2.3. Modify a resource

To modify an uploaded file, you can modify the corresponding resource and upload the modified file to replace the original file. This topic describes how to modify a resource.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to modify and select **Change**.
7. In the **Change Resource** dialog box, modify the parameters as needed. You can change the value of the **Description** parameter and re-upload a file.
8. Submit the resource.
   i. Click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.
9. (Optional)Publish the resource.
   - If the current project is in **Dev** mode, publish the resource to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in **Basic** mode, you do not need to publish the resource after you submit it.

9.9.3.2.4. Move a resource

This topic describes how to move a resource to another directory.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination directory.
8. Click **OK**.

9.9.3.2.5. Delete a resource

You can delete a resource that is no longer used. This topic describes how to delete a resource.

Prerequisites

Resources are created. For more information, see Create a resource.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Resource Management** section appears.
6. In the **Resource Management** section, move the pointer over the icon next to the resource that you want to delete and select **Delete**.

> **Note** You cannot delete the resource if it is referenced by a task.

7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

9.9.3.3. Manage stream processing functions

9.9.3.3.1. Create a folder for storing stream processing functions

This topic describes how to create a folder for storing stream processing functions.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Stream Processing Functions** section, click the icon next to Stream Processing Functions.
7. In the **Create Folder** dialog box, enter a folder name and select a folder.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.

Operation	Description
Move a folder	To move a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination folder. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.3.3.2. Create a UDF

Dataphin provides the Stream Processing Functions section on the Develop page. In this section, you can manage functions to be used in stream processing tasks, including system built-in functions and UDFs. System built-in functions cannot be modified. This topic describes how to create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click .
6. Open the **Create Function** dialog box in one of the following ways:
   - In the Stream Processing Functions section, click the icon next to **Stream Processing Functions**.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing** > **Stream Processing Functions**.
   - In the right-side workspace of the **Develop** page, click the icon next to **Stream**

Processing Functions.

7. In the **Create Function** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the function to create. The name can contain letters, digits, and underscores (_), and must start with a letter.
Resource	The resource to be referenced by the function. You can view the resources that are available for the current project in the drop-down list. **Note** ○ If you select multiple resources, they must be of the same type. ○ If you do not have any resources, you must create one. For more information, see Create a resource.
Class	The class of the function to create. You can extract the class from the resources in the MaxCompute computing engine. Example: `test_udf.UDFGETSrcId`.
Type	The type of the function to create. Valid values: **Window**, **Statistic**, **Numeric**, **String**, **DateTime**, **ip address-related function**, **URL**, **Codec**, **Business**, and **Other**.
Function Syntax	The syntax of the function to create. The syntax of a function is the format for referencing the function. Example: `bigintweekday (datetime date)`.
Description	The description of the function to create. Example: `select get_week_date("20170810",0,2), -- Query the date of the Tuesday in the week of August 10, 2017. from cndata.dual`
Select Directory	The folder of the function to create.

8. Submit the UDF.
   i. After you set the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**. The UDF is created and registered with MaxCompute.

Note If the resources referenced by UDFs are updated, submit the UDFs again so that the UDFs registered with MaxCompute are updated.

You can reference the UDF in an SQL statement and execute the statement in a task on the Ad Hoc Query tab to check whether the function meets your expectation. The following SQL statement is used as an example:

```
select
get_week_date("20170810",0,2), -- Query the date of the Tuesday in the week of August 10, 2017.
from  cndata.dual
```

9. (Optional)Publish the UDF.
   - If the current project is in Dev mode, publish the UDF to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the UDF after you submit it.

9.9.3.3.3. Modify a UDF

Dataphin provides the Stream Processing Functions section on the Develop page. In this section, you can manage functions to be used in stream processing tasks, including system built-in functions and UDFs. System built-in functions cannot be modified. This topic describes how to modify a UDF.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Stream Processing Functions** section, move the pointer over the icon next to the UDF that you want to modify and select **Change**.
7. In the **Change Function** dialog box, set the **Resource**, **Class**, **Type**, **Function Syntax**, and **Description** parameters. For more information about the parameter description, see Create a UDF.
8. Submit the UDF.
   i. After you modify the parameters, click **Submit**.
   ii. In the dialog box that appears, enter your comments.
   iii. Click **OK**.

iv. (Optional)Publish the UDF.

- If the current project is in Dev mode, publish the UDF to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the UDF. After you submit a UDF, it can be referenced in stream processing tasks for scheduling.

9.9.3.3.4. Move a UDF

Dataphin provides the Stream Processing Functions section on the Develop page. In this section, you can manage functions to be used in stream processing tasks, including system built-in functions and UDFs. System built-in functions cannot be modified. This topic describes how to move a UDF.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Stream Processing Functions** section, move the pointer over the icon next to the UDF that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
8. Click **OK**.

9.9.3.3.5. Delete a UDF

Dataphin provides the Stream Processing Functions section on the Develop page. In this section, you can manage functions to be used in stream processing tasks, including system built-in functions and UDFs. System built-in functions cannot be modified. This topic describes how to delete a UDF.

Prerequisites

UDFs are created. For more information, see Create a UDF.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left

corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the left-side navigation submenu, click the icon.
6. In the **Stream Processing Functions** section, move the pointer over the icon next to the UDF that you want to delete and select **Delete**.

   > **Note** You cannot delete a UDF that is referenced in a task.

7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

9.9.3.4. Manage stream processing metatables

9.9.3.4.1. Create a folder for storing stream metatables

This topic describes how to create a folder for storing stream metatables.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
6. In the **Stream Metatables** section, click the icon next to Stream Metatables.
7. In the **Create Folder** dialog box, enter a folder name and select a folder.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Stream Metatables section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.

Operation	Description
Move a folder	To move a folder, perform the following steps: i. In the Stream Metatables section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination folder. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Stream Metatables section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.3.4.2. Create a stream metatable

Dataphin provides the Stream Metatables section on the Develop page. In this section, you can manage stream metatables to be used in stream processing tasks, including input tables, output tables, and dimension tables. This topic describes how to create a stream metatable.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Metatables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
3. Open the **Create Stream Metatable** dialog box by using one of the following methods:
   - In the **Stream Metatables** section, click the icon next to Stream Metatables.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing > Stream Metatables**.
   - In the right-side workspace of the **Develop** page, click the icon next to **Stream Metatables**.

4. In the **Create Stream Metatable** dialog box, set the parameters as required.

Parameter	Description
Metatable Name	The name of the stream metatable. **Note** The name must be unique in the project. Otherwise, the stream metatable cannot be published to the production environment.
Data Source	The data source of the stream metatable. The data sources that are connected to Dataphin are displayed in the drop-down list.
Source Table	The source table. If you select a data source of the **ALIYUN_HBASE**, **KAFKA_9_11**, **TABLE_STORE**, or **DRDS** type, enter the name of a source table from the corresponding data source as needed.
Topic	The source topic. If you select a data source of the **DATAHUB** or **LOG_SERVICE** type, select a topic from the drop-down list or enter the name of a topic from the corresponding data source.
Topic	The source topic. If you select a data source of the **ROCKET_MQ** type, enter the name of a topic from the corresponding data source.
Select Directory	The folder for storing the stream metatable.
Description	The description of the stream metatable.

5. Click **OK**.
6. (Optional)Add one or more fields to the stream metatable by using one of the following methods.
   - a. On the configuration tab of the stream metatable, move the pointer over **Add Field** and select **Use SQL Code**.

b. Enter code in the code editor of the **Use SQL Code** dialog box. Sample code:

```
create table dwi_pub_hbd_cate_mtr (
   rowkey  VARCHAR  comment 'rowkey',
   stat_date  VARCHAR  comment 'stat_date',
   keymin  VARCHAR  comment 'keymin',
   PRIMARY KEY(rowkey)
 )
  with (
    type='alihbase',
    diamondGroup='null',
    zkQuorum='hbasemaster74000.sg94.tbsite.net,hbasemaster74001.sg94.tbsite.net,hbase
master74002.sg94.tbsite.net,hbase74000.sg94.tbsite.net,hbase74001.sg94.tbsite.net',
    diamondKey='null',
    zkNodeParent='/group-sg94-lzd-mix',
    columnFamily='info',
    tableName='dwi_pub_hbd_cate_mtr_001',
    stringWriteMod='true'
 );
```

c. Click **OK**.

- a. On the configuration tab of the stream metatable, move the pointer over **Add Field** and select **Add Multiple Fields**.
  b. Enter code in the code editor of the **Add Multiple Fields** dialog box. Sample code:

```
ID, INT, description, false, true
name, INT, description, false, true
```

- a. On the configuration tab of the stream metatable, move the pointer over **Add Field** and select **Add Single Field**.
  b. In the **Add Single Field** dialog box, set the parameters as required.
  c.

Parameter	Description
Name	The name of the field.
Data Type	The data type of the field.
Primary Key	Specifies whether the field is a primary key.
Header	Specifies whether the field is an attribute field of the data source.
Description	The description of the field.

d. Click **OK**.

The following table describes the operations that you can perform on added fields.

Operation	Description
Edit a field	To edit a field, perform the following steps: i. In the field list of the stream metatable configuration page, find the field that you want to edit and click the icon in the **Actions** column. ii. In the Edit dialog box, modify the parameters as required. iii. Click **Finish**.
Delete a field	To delete a field, perform the following steps: i. In the field list of the stream metatable configuration page, find the field that you want to delete and click the icon in the **Actions** column. ii. In the **Information** message, click **OK**.
Sort fields	To sort fields, perform the following steps: i. On the stream metatable configuration page, click **Sort By**. ii. Sort the fields and click **Finish**.
Search for fields	On the stream metatable configuration page, enter a name or a keyword in the search box to search for fields.
Refresh	On the stream metatable configuration page, click the icon to reparse the fields from the stream metatable and refresh the field list.

7. Save and submit the stream metatable.
   i. Click the icon in the upper-right corner of the configuration tab to save the stream metatable.
   ii. Click the icon in the upper-right corner of the configuration tab to submit the stream metatable.
8. (Optional)Publish the stream metatable.
   - If the current project is in Dev mode, publish the stream metatable to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the stream metatable after you submit it.

9.9.3.4.3. Modify a stream metatable

Dataphin provides the Stream Metatables section on the Develop page. In this section, you can manage stream metatables to be used in stream processing tasks, including input tables, output tables, and dimension tables. This topic describes how to modify a stream metatable.

Prerequisites

Stream metatables are created. For more information, see Create a stream metatable.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Metatables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
3. In the **Stream Metatables** section, move the pointer over the icon next to the stream metatable that you want to modify and select **Change**.
4. (Optional)Steal the lock of the stream metatable.
   i. Steal the lock.
      - If the stream metatable is locked by yourself, you do not need to steal the lock.
      - If the stream metatable is locked by another user, click the icon in the upper-right corner to steal the lock.
   ii. After the lock is stolen, go to the next step to modify the metatable information and the fields in the stream metatable.
5. Modify the metatable information.
   i. In the top navigation bar of the configuration tab, click **Metatable Information**.
   ii. In the **Metatable Information** pane, modify the **Datasource**, **Source topic** or **Source table**, **Description**, and **Parameter k-v configuration** parameters.
   iii. Click **OK**.
6. Modify the fields in the stream metatable. For more information, see Create a stream metatable.
7. Save and submit the stream metatable.
   i. Click the icon in the upper-right corner of the configuration tab to save the stream metatable.
   ii. Click the icon in the upper-right corner of the configuration tab to submit the stream metatable.
8. (Optional)Publish the stream metatable.
   - If the current project is in Dev mode, publish the stream metatable to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the stream metatable

after you submit it.

9.9.3.4.4. View the version information about a stream metatable

Dataphin provides the Stream Metatables section on the Develop page. In this section, you can manage stream metatables to be used in stream processing tasks, including input tables, output tables, and dimension tables. This topic describes how to view the version information about a stream metatable.

Prerequisites

Stream metatables are created. For more information, see Create a stream metatable.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Metatables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
3. In the Stream Metatables section, click the stream metatable for which you want to view the version information.
4. In the top navigation bar of the configuration tab that appears, click **Version Details**. The Version Details pane appears.
5. Click the icon in the **Actions** column of a version to view the comparison between the current version and the reference version.
6. Click the icon in the **Actions** column of a version to view the code of the version.

9.9.3.4.5. Move a stream metatable

Dataphin provides the Stream Metatables section on the Develop page. In this section, you can manage stream metatables to be used in stream processing tasks, including input tables, output tables, and dimension tables. This topic describes how to move a stream metatable.

Prerequisites

Stream metatables are created. For more information, see Create a stream metatable.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Stream Metatables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
3. In the **Stream Metatables** section, move the pointer over the icon next to the stream metatable that you want to move and select **Move**.
4. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
5. Click **OK**.

9.9.3.4.6. Delete a stream metatable

Dataphin provides the Stream Metatables section on the Develop page. In this section, you can manage stream metatables to be used in stream processing tasks, including input tables, output tables, and dimension tables. This topic describes how to delete a stream metatable.

Prerequisites

Stream metatables are created. For more information, see Create a stream metatable.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Metatables** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Metatables** section appears.
3. In the **Stream Metatables** section, move the pointer over the icon next to the stream metatable that you want to delete and select **Delete**.

   > **Note** You cannot delete a stream metatable if it is referenced by a stream processing task.

4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

9.9.3.5. Manage stream processing templates

9.9.3.5.1. Create a folder for storing stream processing templates

This topic describes how to create a folder for storing stream processing templates.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.
6. In the **Stream Processing Templates** section, click the icon next to Stream Processing Templates.
7. In the **Create Folder** dialog box, enter a folder name and select a folder.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination folder. iii. Click **OK**.

Operation	Description
Delete a folder	To delete a folder, perform the following steps: i. In the Stream Processing Templates section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.3.5.2. Create a stream processing template

Dataphin provides the Stream Processing Templates section on the Develop page. In this section, you can manage templates that can be used in stream processing tasks to improve the coding efficiency. This topic describes how to create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Templates** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.
3. Open the **Create Stream Processing Template** dialog box by using one of the following methods:
   - In the **Stream Processing Functions** section, click the icon next to Stream Processing Templates.
   - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing > Stream Processing Templates**.
   - In the right-side workspace of the **Develop** page, click the icon next to **Stream Processing Template**.
4. In the **Create Stream Processing Template** dialog box, set the **Template Name**, **Select Directory**, and **Description** parameters.
5. Click **OK**.
6. Configure the stream processing template.
   i. Write the code of the stream processing template.

ii. In the top navigation bar of the configuration tab, click **Template Details**.

iii. In the **Template Details** pane, click **Parse** in the Parameter Configuration section.

iv. Set **Parameter Description** and **Default Value** for the parameters that are used in the stream processing template as required. In the Basic Information section, you can also modify or add a description for the template.

v. Click **OK**.If you reference this template when you create a stream processing task of the FLINK_TEMPLATE_SQL type, you can modify the parameters in the template on the configuration tab of the stream processing task.

7. Save and submit the stream processing template.

   i. Click the icon in the upper-right corner of the configuration tab to save the stream processing template.

   ii. Click the icon in the upper-right corner of the configuration tab to submit the stream processing template.

8. (Optional)Publish the stream processing template.
   - If the current project is in Dev mode, publish the stream processing template to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the stream processing template after you submit it.

9.9.3.5.3. Modify a stream processing template

Dataphin provides the Stream Processing Templates section on the Develop page. In this section, you can manage templates that can be used in stream processing tasks to improve the coding efficiency. This topic describes how to modify a stream processing template.

Prerequisites

Stream processing templates are created. For more information, see Create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Templates** section.

   i. On the Dataphin homepage, click **R&D** in the top navigation bar.

   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.

   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.

   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.

3. In the **Stream Processing Templates** section, move the pointer over the icon next to the stream processing template that you want to modify and select **Change**.

4. (Optional)Steal the lock of the stream processing template.
   i. Steal the lock.
      - If the stream processing template is locked by yourself, you do not need to steal the lock.
      - If the stream processing template is locked by another user, click the icon in the upper-right corner to steal the lock.
   ii. After the lock is stolen, go to the next step to modify the code and parameters in the stream processing template
5. Modify the code and parameters in the stream processing template. For more information, see Create a stream processing template.
6. Save and submit the stream processing template.
   i. Click the icon in the upper-right corner of the configuration tab to save the stream processing template.
   ii. Click the icon in the upper-right corner of the configuration tab to submit the stream processing template.
7. (Optional)Publish the stream processing template.
   - If the current project is in Dev mode, publish the stream processing template to the corresponding project in Prod mode. For more information, see Publishing management.
   - If the current project is in Basic mode, you do not need to publish the stream processing template after you submit it.

9.9.3.5.4. View the version information about a stream processing template

Dataphin provides the Stream Processing Templates section on the Develop page. In this section, you can manage templates that can be used in stream processing tasks to improve the coding efficiency. This topic describes how to view the version information about a stream processing template.

Prerequisites

Stream processing templates are created. For more information, see Create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Templates** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.

iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.

3. In the Stream Processing Templates section, click the stream processing template for which you want to view the version information.
4. In the top navigation bar of the configuration tab that appears, click **Version Details**. The Version Details pane appears.
5. Click the icon in the **Actions** column of a version to view the comparison between the current version and the reference version.
   - a. In the **Version Comparison** dialog box, click the **Code Comparison** tab to view the comparison between the code in the **Current Version** and **Reference Version** sections.
     b. To roll back to the reference version, click **Roll Back to Reference Version**.
     c. In the **Tip** message, click **OK**.
   - a. In the **Version Comparison** dialog box, click the **Template Parameter Comparison** tab to view the comparison between the parameter information in the **Current Version** and **Reference Version** sections.
     b. To roll back to the reference version, click **Roll Back to Reference Version**.
     c. In the **Tip** message, click **OK**.
6. Click the icon in the **Actions** column of a version to view the code of the version.

9.9.3.5.5. Move a stream processing template

Dataphin provides the Stream Processing Templates section on the Develop page. In this section, you can manage templates that can be used in stream processing tasks to improve the coding efficiency. This topic describes how to move a stream processing template.

Prerequisites

Stream processing templates are created. For more information, see Create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Templates** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.
3. In the **Stream Processing Templates** section, move the pointer over the icon next to the stream processing template that you want to move and select **Move**.

4. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
5. Click **OK**.

9.9.3.5.6. Delete a stream processing template

Dataphin provides the Stream Processing Templates section on the Develop page. In this section, you can manage templates that can be used in stream processing tasks to improve the coding efficiency. This topic describes how to delete a stream processing template.

Prerequisites

Stream processing templates are created. For more information, see Create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Templates** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Templates** section appears.
3. Move the pointer over the icon next to the stream processing template that you want to delete and select **Delete**.

   > **Note** You can delete only the folders that do not contain subfolders or items.

4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

9.9.3.6. Manage stream processing tasks

9.9.3.6.1. Create a folder for storing stream processing tasks

This topic describes how to create a folder for storing stream processing tasks.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left

corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.

4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
6. In the **Stream Processing Tasks** section, click the icon next to Stream Processing Tasks.
7. In the **Create Folder** dialog box, enter a folder name and select a folder.
8. Click **OK**.The following table describes the operations that you can perform on created folders.

Operation	Description
Rename a folder	To rename a folder, perform the following steps: i. In the Stream Processing Tasks section, move the pointer over the icon next to the folder that you want to rename and select **Rename**. ii. In the field that appears, enter a new name. iii. Press the Enter key.
Move a folder	To move a folder, perform the following steps: i. In the Stream Processing Tasks section, move the pointer over the icon next to the folder that you want to move and select **Move**. ii. In the **Move Folder** dialog box, set the **Select Directory** parameter to the destination folder. iii. Click **OK**.
Delete a folder	To delete a folder, perform the following steps: i. In the Stream Processing Tasks section, move the pointer over the icon next to the folder that you want to delete. ii. Select **Delete**. **Note** You can delete only the folders that do not contain subfolders or items.

9.9.3.6.2. Create a stream processing task of the FLINK_SQL type

Dataphin allows you to manage stream processing tasks of the FLINK_SQL and FLINK_TEMPLATE_SQL types. This topic describes how to create a stream processing task of the FLINK_SQL type.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
3. Create a stream processing task of the FLINK_SQL type.
   i. Open the **Create Flink_SQL** dialog box by using one of the following methods:
      - In the Stream Processing Tasks section, click the icon next to **Stream Processing Tasks** and select **FLINK_SQL**.
      - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing** > **Stream Processing Tasks** > **FLINK_SQL**.
      - In the right-side workspace of the **Develop** page, click the icon next to **FLINK_SQL**.
   ii. In the **Create Flink_SQL** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the stream processing task. **Note** The name must be unique in the project. Otherwise, the stream processing task cannot be published to the production environment.
Select Directory	The folder for storing the stream processing task.
Resource Queue	The resource queue of the computing engine for stream processing that is bound to the current project.
Engine Version	The version of the computing engine supported by the selected resource queue.

   iii. Click **OK**.
4. Write the code of the stream processing task.

i. On the configuration tab, write the code to configure the stream processing task.

> **Note** If you do not have permissions to access the tables referenced in the stream processing task, enter related SET statements at the beginning of the code to specify the AccessKey ID and AccessKey secret required for accessing the tables.

ii. Click **Precompile** to check the syntax and related permissions of the stream processing task.

iii. Click **Debugging**. Then, you can configure the sample data for debugging and locally debug the stream processing task to make sure that the code to be submitted is correct.

iv. In the **Configure Debugging Data** dialog box, select a table in the left-side table list and click **Auto Sampling**.

v. In the pop-up dialog box, set the **Data Records** parameter and click **OK**.

vi. After the sampling is complete for all tables, click **OK** in the lower-right corner of the **Configure Debugging Data** dialog box.

vii. View the debugging data, intermediate result, and debugging result on the **Result** tab.

5. Set the parameters of the stream processing task.
   i. In the top navigation bar of the configuration tab, click **Task Parameters**.
   ii. In the Task Parameters pane, set parameters, including the checkpoint mode and interval, based on the sample code of the stream processing task.
   iii. Click **OK**.
6. Configure the stream processing task.
   i. In the top navigation bar of the configuration tab, click **Task Configuration**. The **Task Configuration** pane appears.

ii. Set the **Resource Configuration** parameter.

- If you select **System Recommended**, you do not need to configure resources.
- If you select **Custom**, you must click **Configure** to configure resources. You can configure resources in one of the following modes: **Visualization Mode** and **Code Editor**.

Mode	Description
Visualization Mode	To configure resources in visualization mode, perform the following steps: a. On the resource configuration page, click **Visualization Mode**. b. Click the icon in the upper-right corner of a compute node. c. In the dialog box that appears, set the parameters as required. d. Click **OK**. e. After you set the execution parameters for all compute nodes, click **Save** in the upper-right corner of the page.
Code Editor	To configure resources in the code editor, perform the following steps: a. On the resource configuration page, click **Code Editor**. b. In the code editor, set the execution parameters for compute nodes as required. Then, click **Save** in the upper-right corner of the page.

iii. In the **Task Configuration** pane, click the icon to turn on **Auto Optimization** and set the **Maximum CUs** and **Expected Maximum Memory** parameters as required. Then, click **OK**.

7. Save and submit the stream processing task.

i. Click the icon in the upper-right corner of the configuration tab to save the stream processing task.

ii. Click the icon in the upper-right corner of the configuration tab to submit the stream processing task.

8. (Optional)Publish the stream processing task.

- If the current project is in Dev mode, publish the stream processing task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the stream processing task after you submit it.

9.9.3.6.3. Create a stream processing task of the FLINK_TEMPLATE_SQL type

Dataphin allows you to manage stream processing tasks of the FLINK_SQL and FLINK_TEMPLATE_SQL types. This topic describes how to create a stream processing task of the FLINK_TEMPLATE_SQL type.

Prerequisites

Stream processing templates are created. For more information, see Create a stream processing template.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Stream Processing Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
   iii. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
   iv. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
3. Create a stream processing task of the FLINK_TEMPLATE_SQL type.
   i. Open the **Create Flink_Template_SQL** dialog box by using one of the following methods:
      - In the Stream Processing Tasks section, click the icon next to **Stream Processing Tasks** and select **FLINK_TEMPLATE_SQL**.
      - In the left-side navigation pane, click the icon next to the project name and choose **Data Processing** > **Stream Processing Tasks** > **FLINK_TEMPLATE_SQL**.
      - In the right-side workspace of the **Develop** page, click the icon next to **FLINK_TEMPLATE_SQL**.

ii. In the **Create Flink_Template_SQL** dialog box, set the parameters as required.

Parameter	Description
Name	The name of the stream processing task. **Note** The name must be unique in the project. Otherwise, the stream processing task cannot be published to the production environment.
Select Directory	The folder for storing the stream processing task.
Resource Queue	The resource queue of the computing engine for stream processing that is bound to the current project.
Engine Version	The version of the computing engine supported by the selected resource queue.
Referenced Template	The template referenced by the stream processing task.

iii. Click **OK**.

4. Write the code of the stream processing task.
   i. On the configuration tab, write the code to configure the stream processing task.
   ii. Click **Precompile** to check the syntax and related permissions of the stream processing task.
   iii. Click **Debugging**. Then, you can configure the sample data for debugging and locally debug the stream processing task to make sure that the code to be submitted is correct.
   iv. In the **Configure Debugging Data** dialog box, select a table in the left-side table list and click **Auto Sampling**.
   v. In the pop-up dialog box, set the **Data Records** parameter and click **OK**.
   vi. After the sampling is complete for all tables, click **OK** in the lower-right corner of the **Configure Debugging Data** dialog box.
   vii. View the debugging data, intermediate result, and debugging result on the **Result** tab.
5. Set the parameters of the stream processing task.
   i. In the top navigation bar of the configuration tab, click **Task Parameters**.
   ii. In the Task Parameters pane, set parameters, including the checkpoint mode and interval, based on the sample code of the stream processing task.
   iii. Click **OK**.
6. Configure the stream processing task.
   i. In the top navigation bar of the configuration tab, click **Task Configuration**. The **Task Configuration** pane appears.

ii. Set the **Resource Configuration** parameter.

- If you select **System Recommended**, you do not need to configure resources.
- If you select **Custom**, you must click **Configure** to configure resources. You can configure resources in one of the following modes: **Visualization Mode** and **Code Editor**.

Mode	Description
Visualization Mode	To configure resources in visualization mode, perform the following steps: a. On the resource configuration page, click **Visualization Mode**. b. Click the icon in the upper-right corner of a compute node. c. In the dialog box that appears, set the parameters as required. d. Click **OK**. e. After you set the execution parameters for all compute nodes, click **Save** in the upper-right corner of the page.
Code Editor	To configure resources in the code editor, perform the following steps: a. On the resource configuration page, click **Code Editor**. b. In the code editor, set the execution parameters for compute nodes as required. Then, click **Save** in the upper-right corner of the page.

iii. In the **Task Configuration** pane, click the icon to turn on **Auto Optimization** and set the **Maximum CUs** and **Expected Maximum Memory** parameters as required. Then, click **OK**.

7. Set the parameters in the stream processing template referenced by the stream processing task.
   i. In the top navigation bar of the configuration tab, click **Template Parameters**.
   ii. In the **Template Parameters** pane, set the **Parameter Value** parameter.
   iii. Click **OK**.
8. Save and submit the stream processing task.
   i. Click the icon in the upper-right corner of the configuration tab to save the stream processing task.
   ii. Click the icon in the upper-right corner of the configuration tab to submit the stream processing task.

iii. (Optional)Publish the stream processing task.

- If the current project is in Dev mode, publish the stream processing task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the stream processing task after you submit it.

9.9.3.6.4. Modify a stream processing task

Stream processing tasks are used to process data in real time during data development. This topic describes how to modify a stream processing task.

Prerequisites

Stream processing tasks are created. For more information, see Create a stream processing task of the FLINK_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
6. In the **Stream Processing Tasks** section, move the pointer over the icon next to the stream processing task that you want to modify and select **Change**.
7. (Optional)Steal the lock of the stream processing task.
   i. Steal the lock.
      - If the stream processing task is locked by yourself, you do not need to steal the lock.
      - If the stream processing task is locked by another user, click the icon in the upper-right corner to steal the lock.
   ii. After the lock is stolen, go to the next step to modify the code, parameters, and configuration of the stream processing task.
8. Modify the code, parameters, and configuration of the stream processing task. For more information, see Create a stream processing task of the FLINK_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.
9. Save and submit the stream processing task.
   i. Click the icon in the upper-right corner of the configuration tab to save the stream processing task.

ii. Click the icon in the upper-right corner of the configuration tab to submit the stream processing task.

iii. (Optional)Publish the stream processing task.

- If the current project is in Dev mode, publish the stream processing task to the corresponding project in Prod mode. For more information, see Publishing management.
- If the current project is in Basic mode, you do not need to publish the stream processing task after you submit it.

9.9.3.6.5. View the version information about a stream processing task

This topic describes how to view the version information about a stream processing task.

Prerequisites

Stream processing tasks are created. For more information, see Create a stream processing task of the FLINK_TEMPLATE_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
6. In the **Stream Processing Tasks** section, click the stream processing task for which you want to view the version information.
7. In the top navigation bar of the configuration tab that appears, click **Version Details**. The Version Details pane appears.
8. Click the icon in the **Actions** column of a version to view the comparison between the current version and the reference version.
   - a. In the **Version Comparison** dialog box, click the **Code Comparison** tab to view the comparison between the code in the **Current Version** and **Reference Version** sections.
     b. To roll back to the reference version, click **Roll Back to Reference Version**.
     c. In the **Tip** message, click **OK**.
   - a. In the **Version Comparison** dialog box, click the **Task Parameter Comparison** tab to view the comparison between the parameter information in the **Current Version** and **Reference Version** sections.

b. To roll back to the reference version, click **Roll Back to Reference Version**.

c. In the **Tip** message, click **OK**.

9. Click the icon in the **Actions** column of a version to view the code of the version.

9.9.3.6.6. Move a stream processing task

Stream processing tasks are used to process data in real time during data development. This topic describes how to move a stream processing task.

Prerequisites

Stream processing tasks are created. For more information, see Create a stream processing task of the FLINK_TEMPLATE_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode. You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
6. In the **Stream Processing Tasks** section, move the pointer over the icon next to the stream processing task that you want to move and select **Move**.
7. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
8. Click **OK**.

9.9.3.6.7. Unpublish or delete a stream processing task

This topic describes how to unpublish, unpublish and delete, and delete stream processing tasks in different states.

Prerequisites

Stream processing tasks are created. For more information, see Create a stream processing task of the FLINK_TEMPLATE_SQL type or Create a stream processing task of the FLINK_TEMPLATE_SQL type.

Context

- A stream processing task may be in the following states:
  - After you create and save a stream processing task, it enters the **Draft** state.
  - After you submit a stream processing task, it enters the **Submitted** state.

  - After you modify and save a stream processing task in the **Submitted** state, it enters the **Developing** state.
  - After you unpublish a stream processing task in the **Submitted** state, it enters the **Draft** state.

- You can unpublish stream processing tasks only when they are in the **Developing** or **Submitted** state.
- You can delete stream processing tasks only when they are in the **Draft**, **Developing** or **Submitted** state.

Unpublish a stream processing task

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. (Optional)On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Data Processing** tab in the left-side navigation pane.
5. On the Data Processing tab, move the pointer over the left-side navigation submenu and click the icon. The **Steam Processing Tasks** section appears.
6. In the **Stream Processing Tasks** section, move the pointer over the icon next to the stream processing task that you want to unpublish and select **Unpublish**.
7. In the **Tip** dialog box, enter your comments.
8. Click **OK**.

Delete a stream processing task

1. In the **Stream Processing Tasks** section, move the pointer over the icon next to the stream processing task that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

9.9.4. Ad hoc query

9.9.4.1. Overview

Dataphin provides the ad hoc query feature for you to query data. Dataphin can automatically identify and change the SQL syntax based on your computing engine type.

Dataphin uses a business unit as the namespace for its unique logical table models. Therefore, Dataphin uses the following SQL syntax for logical tables:

- Logical tables are referenced in SQL statements in the format of Business unit.Logical table.
- The query logic can be in the format of [Business unit.Logical table.Dimension-associated role. ···. Dimension-associated role.Logical dimension table field]. If the query logic is used as a condition of a SELECT statement or a WHERE clause, you only need to enter the logic in the format of [Logical table.Dimension-associated role. ···.Dimension-associated role.Logical

dimension table field] to search for fields associated with dimensions.

Note The primary key field of a virtual dimension cannot be used in SQL statements. A virtual dimension can be used only to construct a statistic granularity.

9.9.4.2. Create an ad hoc query task

This topic describes how to create an ad hoc query task.

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.

   You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.
5. Create a folder for storing ad hoc query tasks.
   i. On the **Ad Hoc Query** tab, click the icon next to **Ad Hoc Query**.
   ii. In the **Create Folder** dialog box, enter a folder name and select a folder.
   iii. Click **OK**.
6. Create an ad hoc query task.
   i. Open the **Create Item** dialog box by using one of the following methods:
      - In the left-side navigation pane, click the icon next to the project name and select **Ad Hoc Query**.
      - On the **Ad Hoc Query** tab, click the icon next to Ad Hoc Query.
      - In the right-side workspace of the **Develop** page, click the icon next to **SQL Query** in the **Ad Hoc Query** section.
   ii. In the **Create Item** dialog box, set the **Name** and **Description** parameters and select the created folder.
   iii. Click **OK**. The Code Editor tab appears.
7. Write the code of the ad hoc query task on the Code Editor tab.
   i. Write SQL statements to create an ad hoc query task.
   ii. Click **Precompile** in the upper-right corner of the Code Editor tab to check whether the SQL statements comply with the standard.

   If the SQL statements do not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statements so that they comply with the standard.
   iii. Click **Run** in the upper-right corner of the Code Editor tab to execute the SQL statements. Then, check whether the SQL statements are executed based on the information on the Console tab in the lower part of the page.
8. If the SQL statements are executed, click the icon in the upper-right corner of the Code

Editor tab.

9.9.4.3. Manage ad hoc query tasks

This topic describes how to edit, run, delete, move, and rename an ad hoc query task.

Edit an ad hoc query task

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Dev** or **Basic** mode.

   You can skip this step if the current project is in **Dev** or **Basic** mode.
4. On the **Develop** page, click the **Ad Hoc Query** tab in the left-side navigation pane.
5. On the **Ad Hoc Query** tab, move the pointer over the icon next to the ad hoc query task that you want to edit and select **Change**.
6. (Optional)In the upper-right corner of the configuration tab that appears, click the icon to steal the lock for the ad hoc query task.

   If you need to edit an ad hoc query task that is created by another user or an ad hoc query task that is created by yourself but has been locked by another user, you must steal the lock before you can edit the ad hoc query task.
   - If the ad hoc query task is created by another user or the ad hoc query task is created by yourself but has been locked by another user, you must steal the lock before you can edit the ad hoc query task.
   - If the ad hoc query task is locked by yourself, you do not need to steal the lock.

7. i. In the code editor on the configuration tab, edit the SQL statements of the ad hoc query task.

   In the SQL statements, reference tables based on the following rules:

   - When you query data in a physical table in another project, prefix the table name with the corresponding project name in SQL statements. For example, in the `select * from cloudtest_dev.table4` statement, cloudtest_dev indicates the project name and table4 indicates the name of the physical table.
   - When you query data in a logical table, prefix the table name with the corresponding business unit name in SQL statements. For example, in the `select province from ld_practice.dim_province` statement, ld_practice indicates the business unit name and dim_province indicates the name of the logical table.
   - When you query data in the development environment, suffix the name of the business unit or project in Prod mode with **_dev**. Dataphin automatically generates the corresponding variable for the business unit or project in Prod mode. For example, if you reference a business unit named LD_Trade, Dataphin automatically generates the business unit variable ${LD_Trade}. By default, this variable is replaced with LD_Trade_dev when the SQL statements are executed in the development environment, and replaced with LD_Trade when the SQL statements are executed in the production environment. You can assign a specific value for the variable when you write the SQL statements. This variable helps improve the flexibility of the SQL statements when they are executed in different environments.

   ii. Click **Precompile** in the upper-right corner of the configuration tab to check whether the SQL statements comply with the standard.

   If the SQL statements do not comply with the standard, click **Beautify** in the upper-right corner. Then, Dataphin automatically adjusts the SQL statements so that they comply with the standard.

   iii. Click **Run** in the upper-right corner of the configuration tab to execute the SQL statements. Then, check whether the SQL statements are executed based on the information on the Console tab in the lower part of the page.

8. If the SQL statements are executed, click the icon in the upper-right corner of the configuration tab.

Delete an ad hoc query task

> **Note** You can delete only ad hoc query tasks that have been locked by yourself.

1. On the **Ad Hoc Query** tab, move the pointer over the icon next to the ad hoc query task that you want to delete and select **Delete**.
2. In the **Tip** dialog box, enter your comments.
3. Click **OK**.

Move an ad hoc query task

> **Note** You can move only ad hoc query tasks that have been locked by yourself.

1. On the **Ad Hoc Query** tab, move the pointer over the icon next to the ad hoc query task that you want to move and select **Move**.
2. In the **Move File** dialog box, set the **Select Directory** parameter to the destination folder.
3. Click **OK**.

Rename an ad hoc query task

> **Note** You can rename only ad hoc query tasks that have been locked by yourself.

1. On the **Ad Hoc Query** tab, move the pointer over the icon next to the ad hoc query task that you want to rename and select **Rename**.
2. In the field that appears, enter a new name.
3. Press the Enter key.

9.10. Data distilling

9.10.1. Overview

Based on the data accumulated in data modeling of a Data Mid-End, the data distilling module of Dataphin provides data correlation and in-depth data mining for you to extract value from the data of target objects. This module generates code and scheduling tasks, identifies and connects target entities, and then generates tags for the target entities. It can quickly be applied to various businesses.

Before you use the data distilling module, follow these steps to initialize the module:

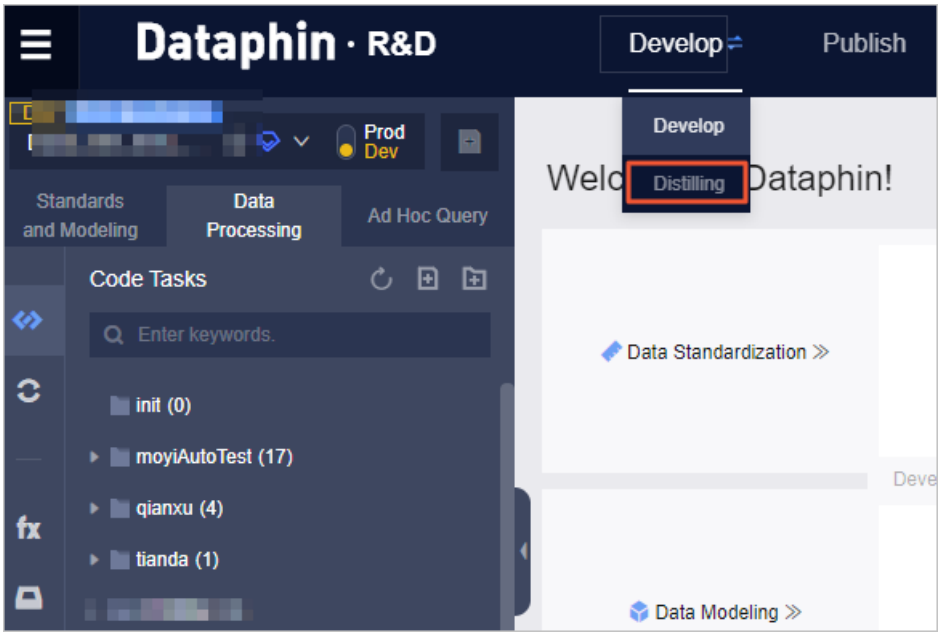
1. Log on to the Dataphin console.
2. Go to the R&D page. In the top navigation bar, move the pointer over **Develop** and select **Distilling**. The **Initialization Configuration** page appears.
3. Select a computing engine from the **Computing Engine** drop-down list and click **Test Connectivity**.

   If no computing engine is available, create one as described in Computing engines.
4. After the computing engine passes the connectivity test, click **Confirm and Initialize**. After the initialization is completed, go to the Distilling tab to develop data.

The data distilling module identifies and associates the master data in the Data Mid-End, that is, the core objects throughout all isolated business domains. Then, this module can interconnect data silos and further extract value from the data of target objects such as high-value tags that can be directly used. In this way, this module accumulates data assets to build a data distilling center.

The data distilling module consists of the Behavior Engine and Tag Engine: The Behavior Engine allows you to configure detailed behaviors and displays behavior statistics. The Tag Engine provides a graphical configuration workbench. It allows you to configure and automatically generate tags and manage tag tasks in different development states.

On the Dataphin homepage, click **R&D** in the top navigation bar. On the R&D page that appears, move the pointer over **Develop** in the top navigation bar and select **Distilling**. The Distilling tab appears.



9.10.2. Behavior Engine

9.10.2.1. Overview

This topic describes the features of the Behavior Engine.

The Behavior Engine consists of the Behavioral Elements, Behavior Rule, and Dashboard submodules.

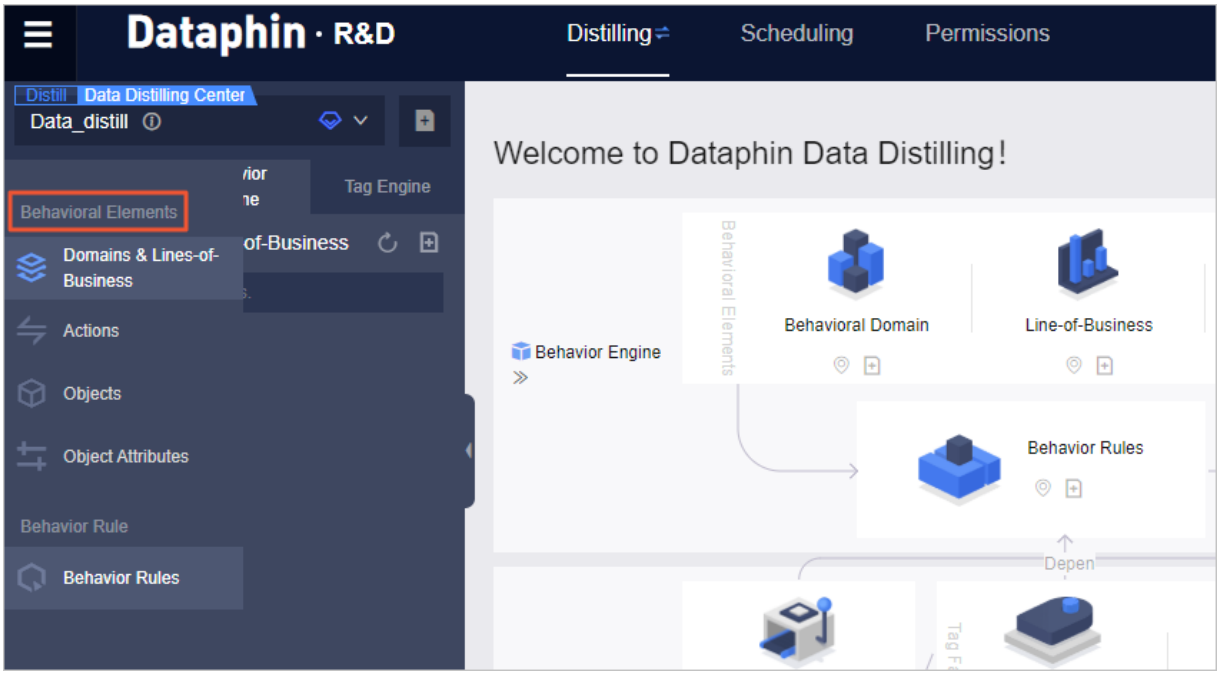
Behavioral Elements submodule

The Behavioral Elements submodule defines standard behavioral elements and their categories. Behavioral elements are categorized into behavioral domains, lines-of-business, actions, objects, and object attributes. You can manage behavioral elements by category.

- Behavioral domain: aggregates behavioral data that has consistent business meanings, such as the e-commerce domain and the entertainment domain.
- Line-of-business: segments behavioral data based on the behavioral domain, such as the Taobao line-of-business and the Tmall line-of-business. Lines-of-business are independent of each other.
- Action: the operation that the behavior subject performs, such as purchasing and browsing.
- Object: the thing on which the behavior subject performs the action, such as a commodity or a movie.
- Object attribute: the description of the object, such as the name, brand, and year.

For example, you can abstract the e-commerce industry as a behavioral domain, and divide the behavioral data in the e-commerce domain into the line-of-business in Mainland China and the line-of-business outside Mainland China by region. The action is payment and the object is commodity. Object attributes include the category, brand, price, size, and color that describe the commodity.

On the Distilling tab, click **Behavior Engine** in the left-side navigation pane. Then, move the pointer over the left-side navigation submenu. You can click a type of behavioral element to manage them on the management page.

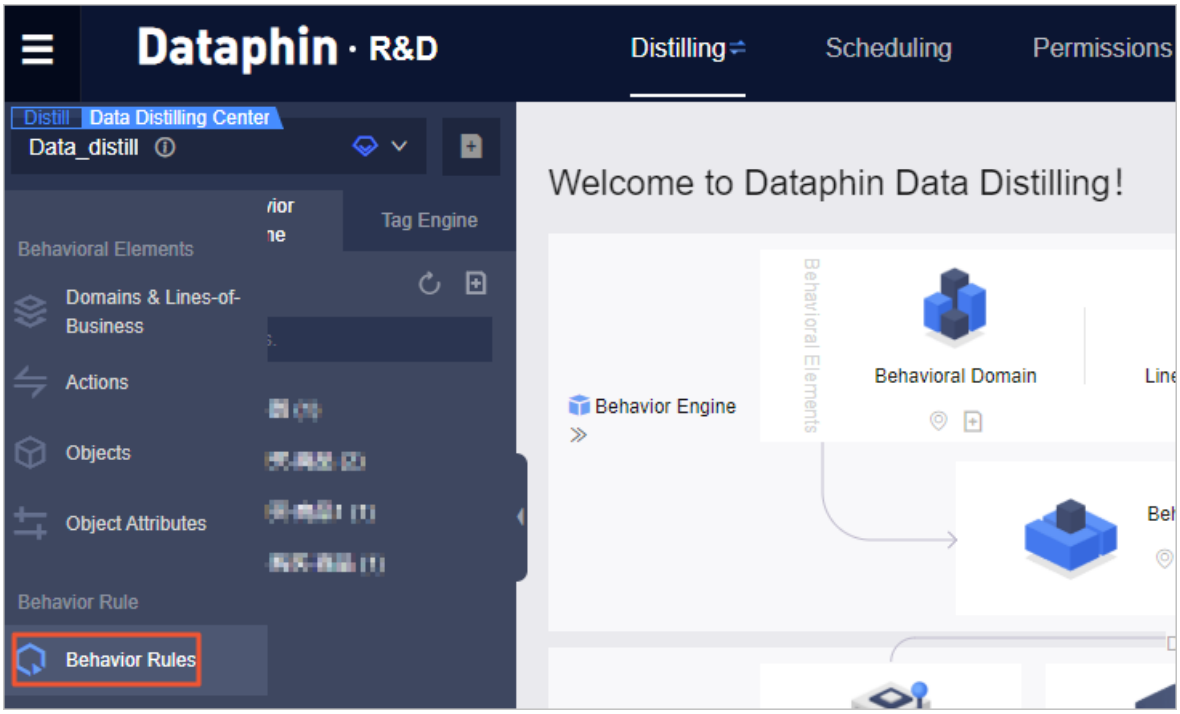


Behavior Rule submodule

Based on the defined behavioral elements, you can configure the corresponding data source and data processing and cleansing rules. Dataphin provides a data source rule for each behavior that is based on a behavioral domain, a line-of-business, an action, and an object to match standardized and structured behaviors with the actual data.

A behavior rule is determined based on the unique behavior and source table. When you preview, save, or submit a behavior rule, Dataphin checks whether the specific behavior and source table are unique.

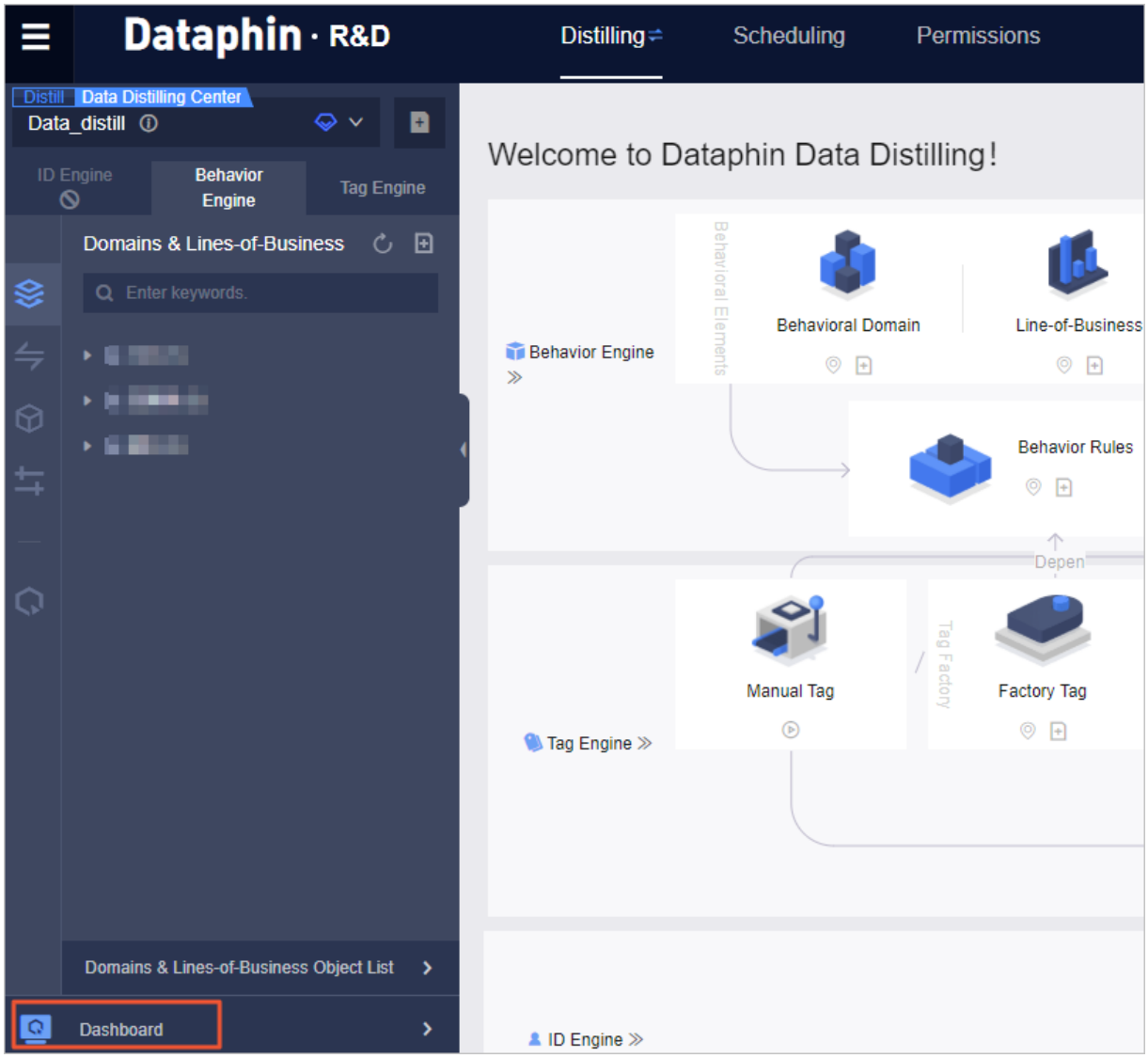
On the Distilling tab, click **Behavior Engine** in the left-side navigation pane. Then, move the pointer over the left-side navigation submenu. You can click **Behavior Rules** to go to the management page.



Dashboard submodule

The Dashboard submodule displays the distribution of behavioral data and the sampling data. It helps you dynamically obtain standardized and structured aggregation results of all behaviors centered on target objects.

On the Distilling tab, click **Behavior Engine** in the left-side navigation pane. Then, click **Dashboard** at the bottom. The Dashboard tab appears.

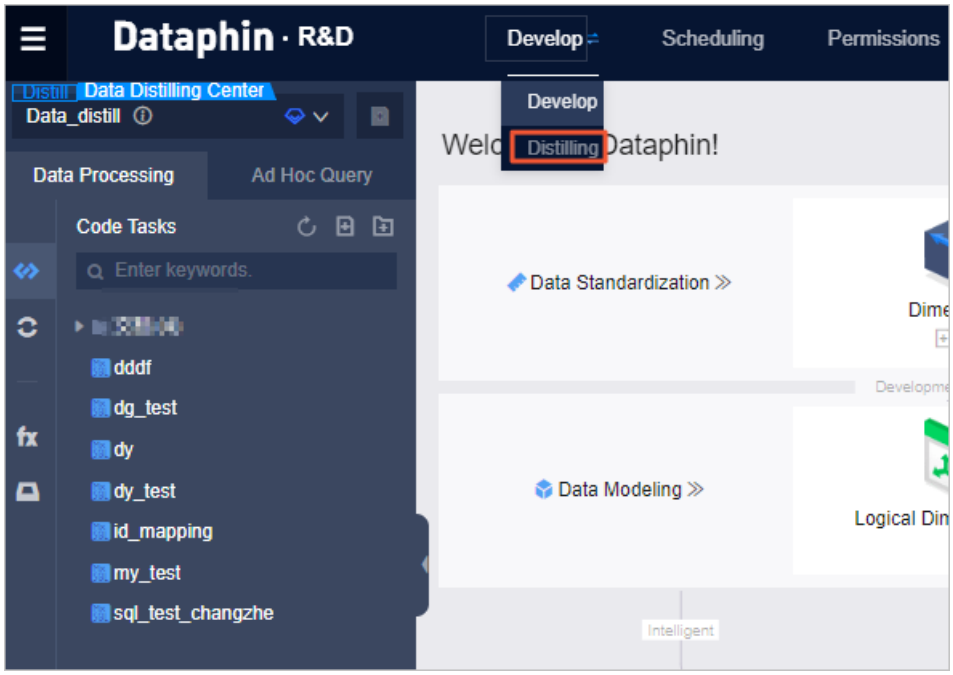


9.10.2.2. Manage behavioral domains

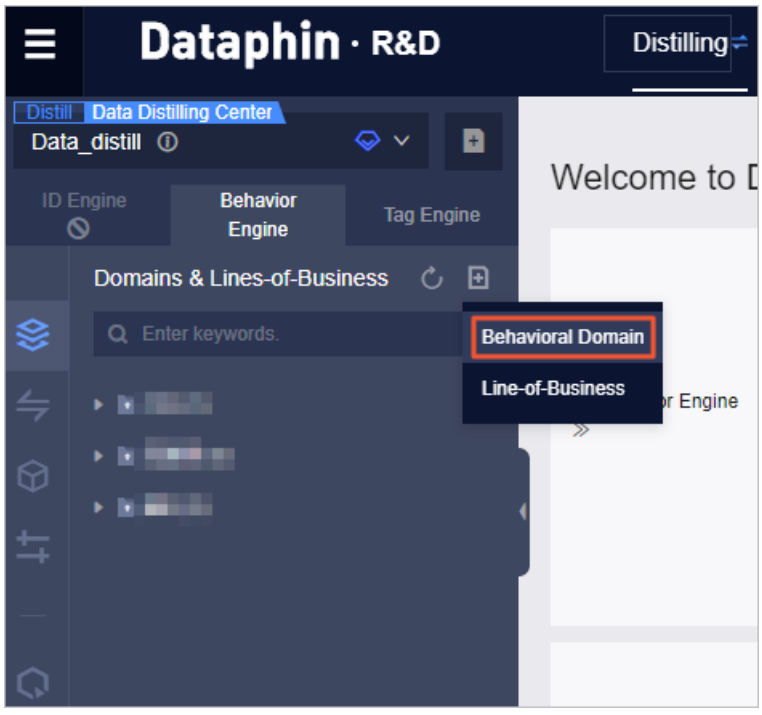
This topic describes how to create, edit, and delete a behavioral domain.

Create a behavioral domain

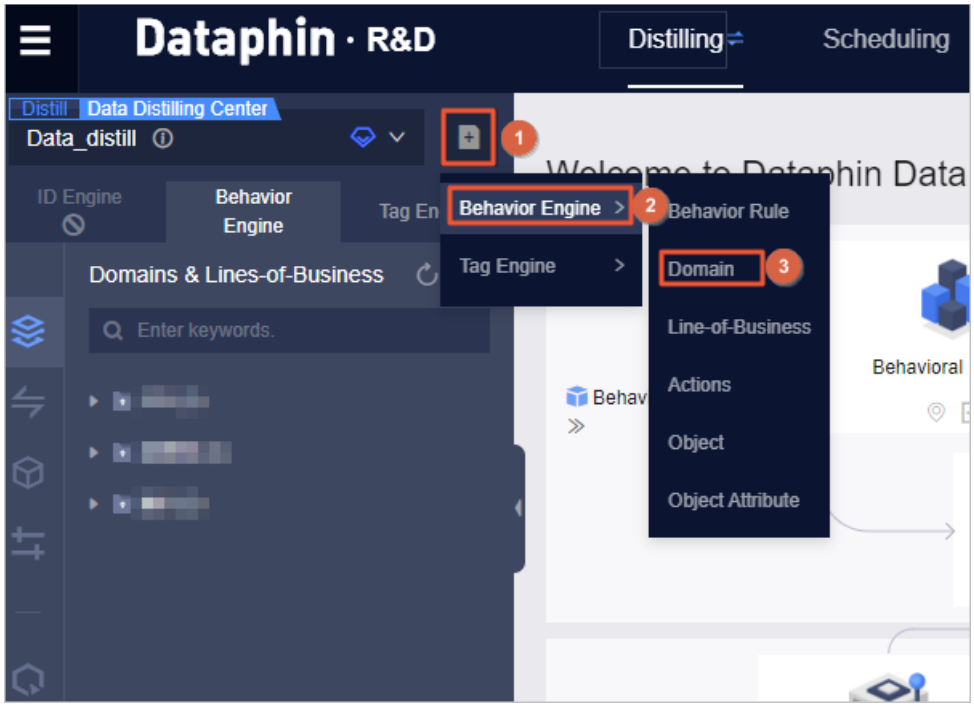
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



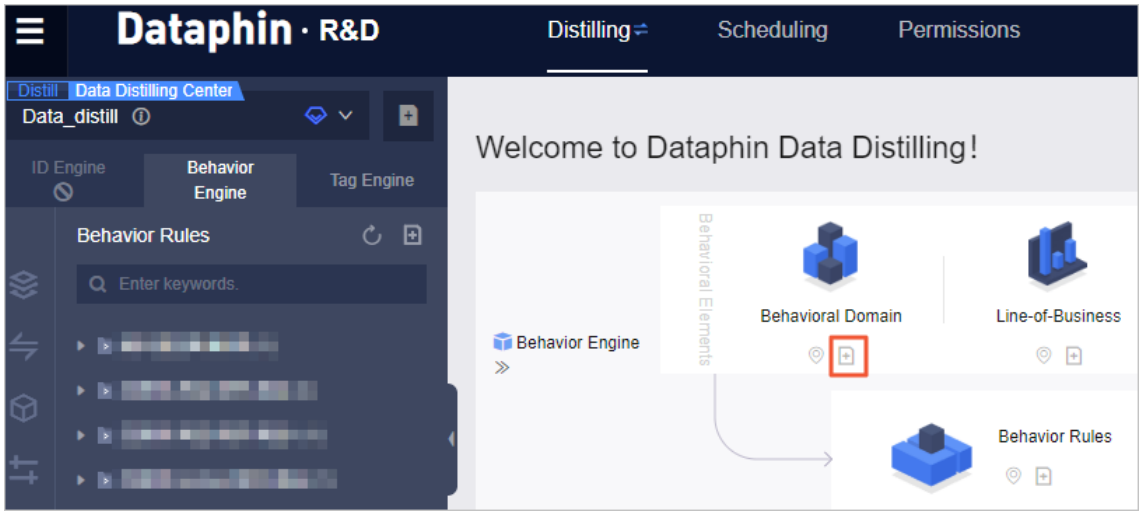
4. Open the **Create Domain** dialog box in one of the following ways:
   - On the Distilling tab, click **Behavior Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon and select **Behavioral Domain**.



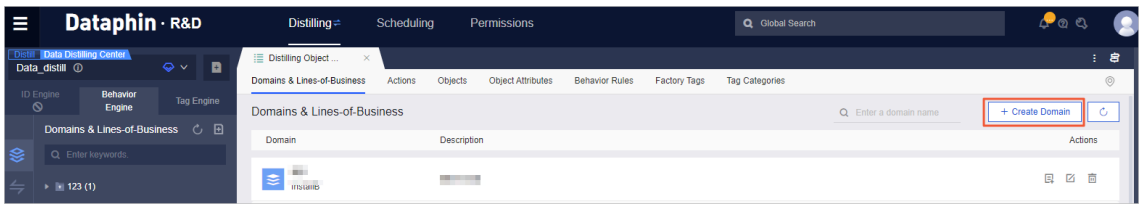
   - On the Distilling tab, click the icon next to the project name and choose **Behavior Engine > Domain**.



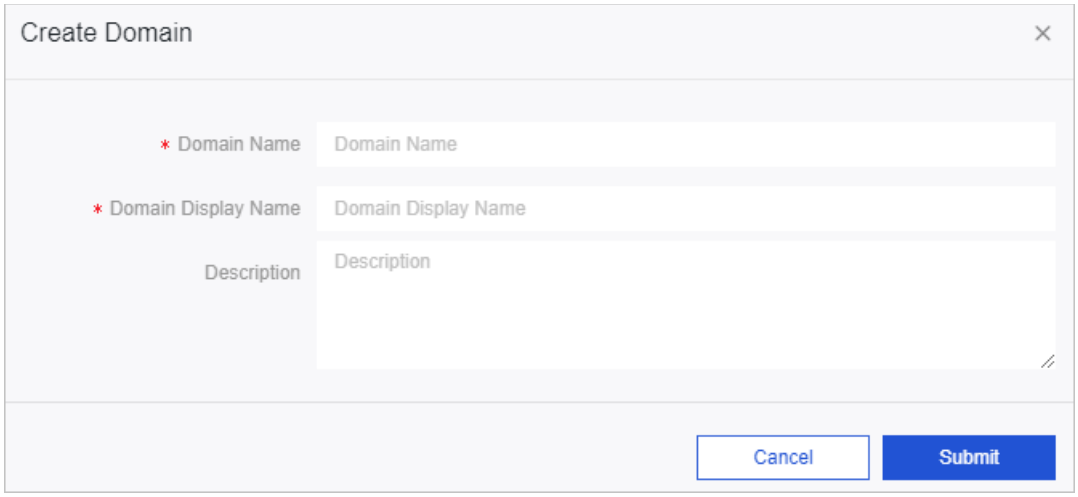
- On the Distilling tab, click the icon below **Behavioral Domain** in the workspace.



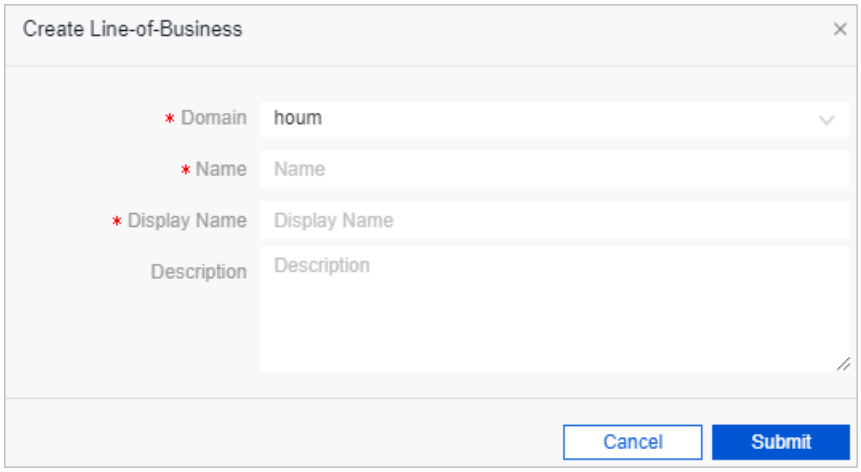
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the **Domains & Lines-of-Business** page that appears in the workspace, click **+ Create Domain** in the upper-right corner.



5. In the Create Domain dialog box that appears, set the **Domain Name**, **Domain Display Name**, and **Description** parameters. Then, click **Submit**.



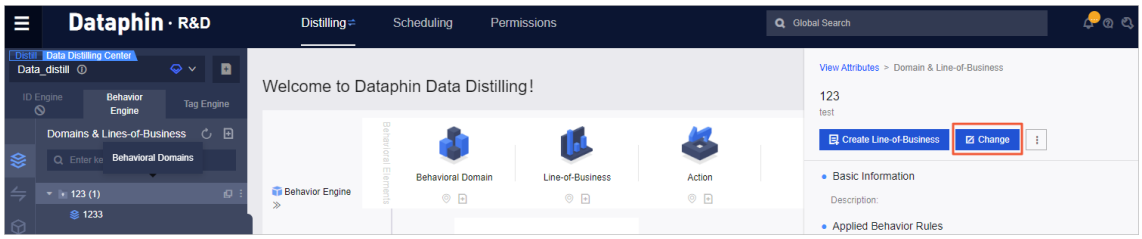
6. In the **Description** dialog box that appears, enter the comments on the behavioral domain to be created. Then, click **OK**.



> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is created. Therefore, you must enter and submit comments when creating a behavioral element.

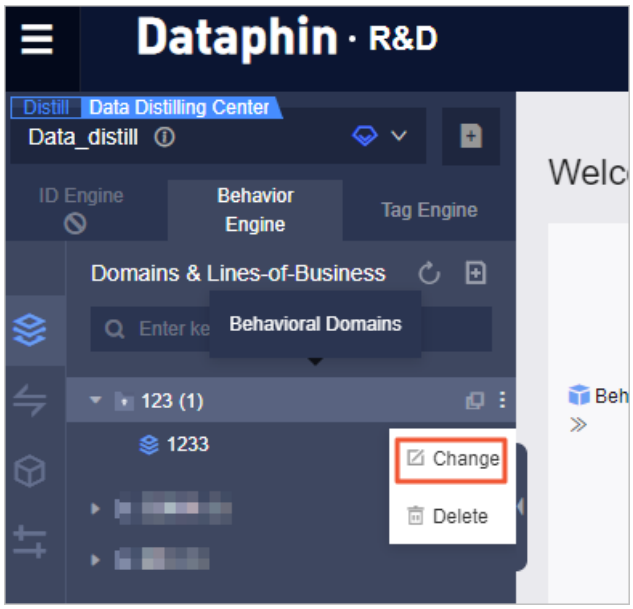
Edit a behavioral domain

1. Open the **Change Domain** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side list, click the target behavioral domain. On the View Attributes tab that appears, click **Change**.

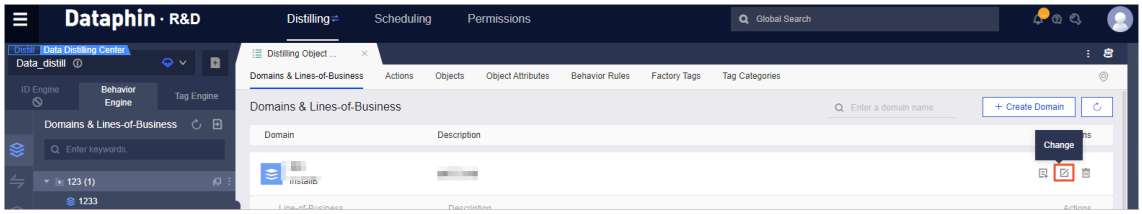


   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the

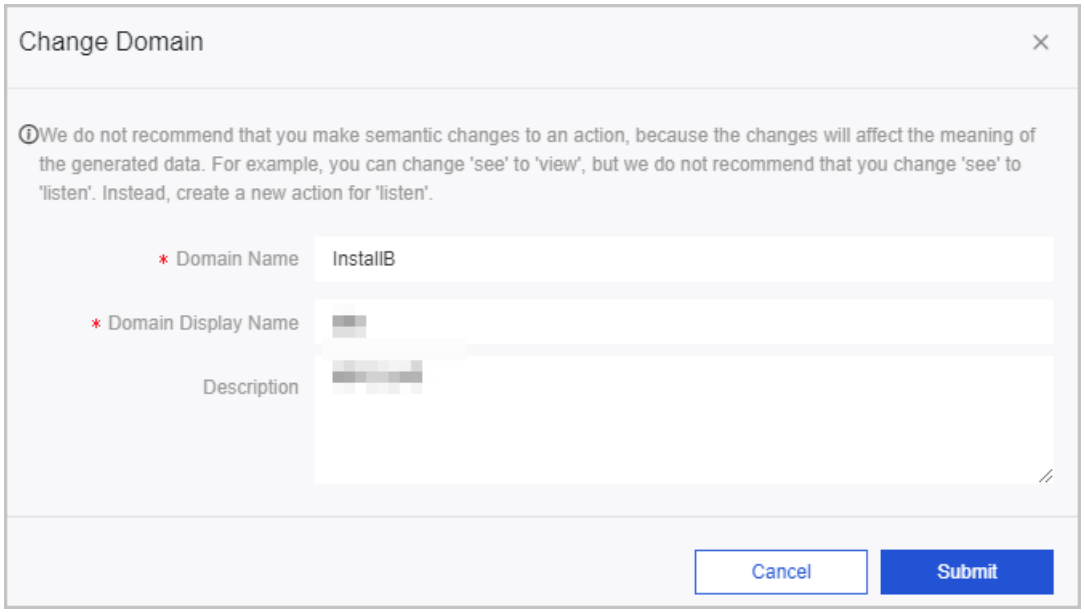
icon on the left-side navigation submenu. In the left-side list, move the pointer over the icon next to the target behavioral domain and select **Change**.



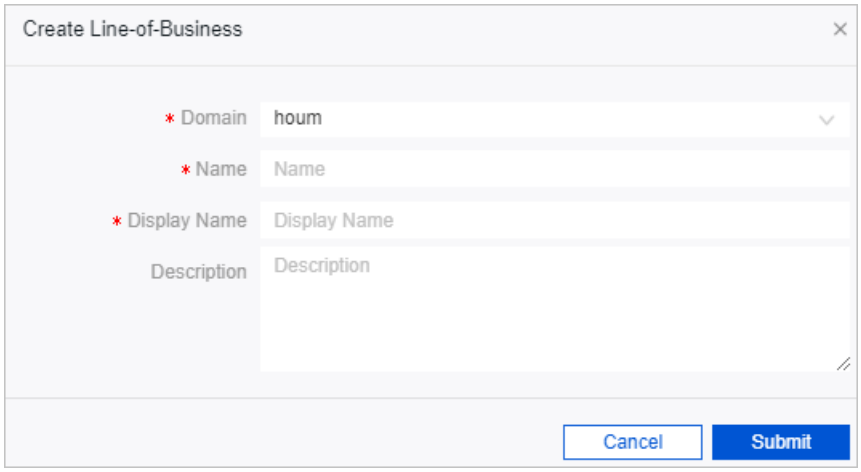
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the **Domains & Lines-of-Business** page that appears in the workspace, click the icon in the **Actions** column of the target behavioral domain.



2. In the Change Domain dialog box that appears, set the **Domain Name**, **Domain Display Name**, and **Description** parameters. Then, click **Submit**.



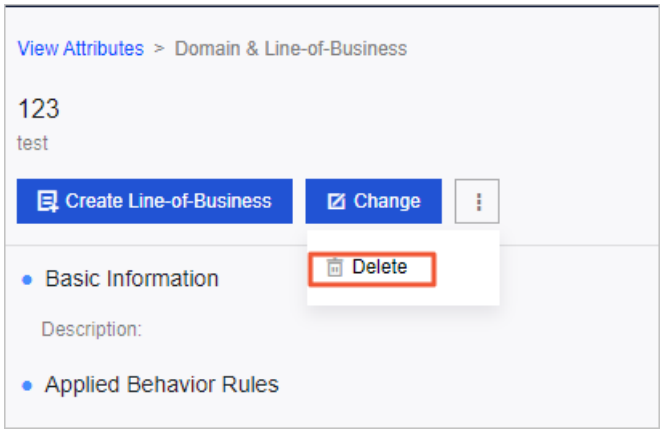
3. In the **Description** dialog box that appears, enter the comments on the behavioral domain to be modified. Then, click **OK**.



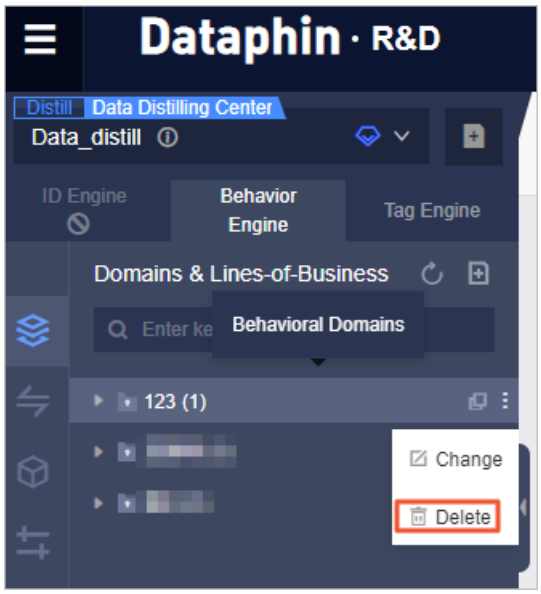
> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is edited. Therefore, you must enter and submit comments when editing a behavioral element.

Delete a behavioral domain

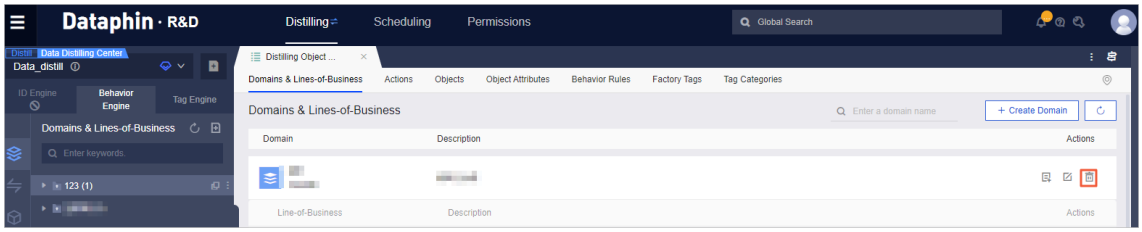
1. Open the **Delete Domain** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side list, click the target behavioral domain. On the View Attributes tab that appears, click the icon and select **Delete**.



- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side list, move the pointer over the icon next to the target behavioral domain and select **Delete**.



- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the **Domains & Lines-of-Business** page that appears in the workspace, click the icon in the **Actions** column of the target behavioral domain.



2. In the **Delete Domain** dialog box that appears, enter the comments on the behavioral domain to be deleted. Then, click **OK**.

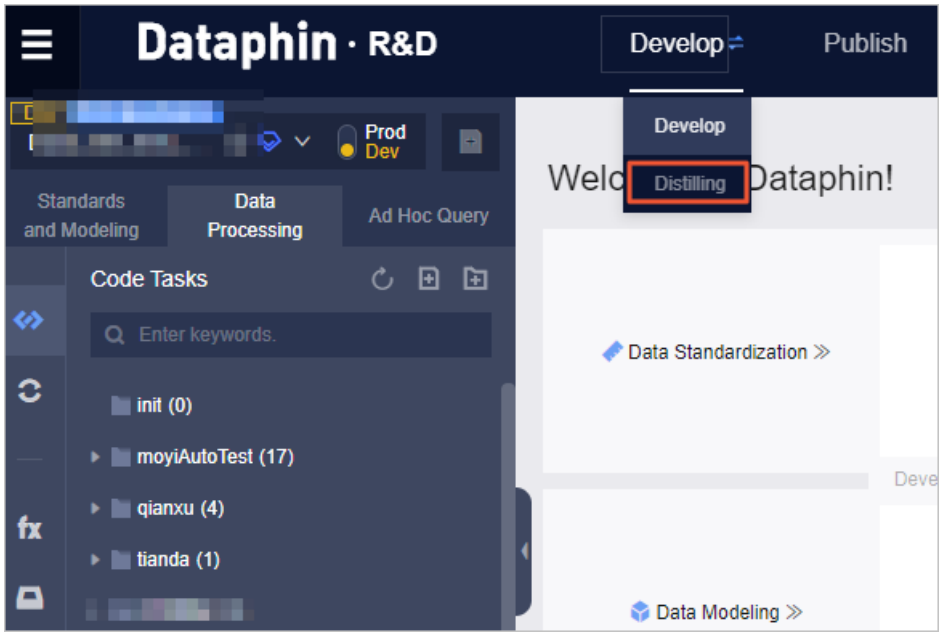
> **Note** A behavioral element is published to the production environment after it is created and is deleted from the production environment after the deletion operation. Therefore, you must enter and submit comments when deleting a behavioral element.

9.10.2.3. Manage lines-of-business

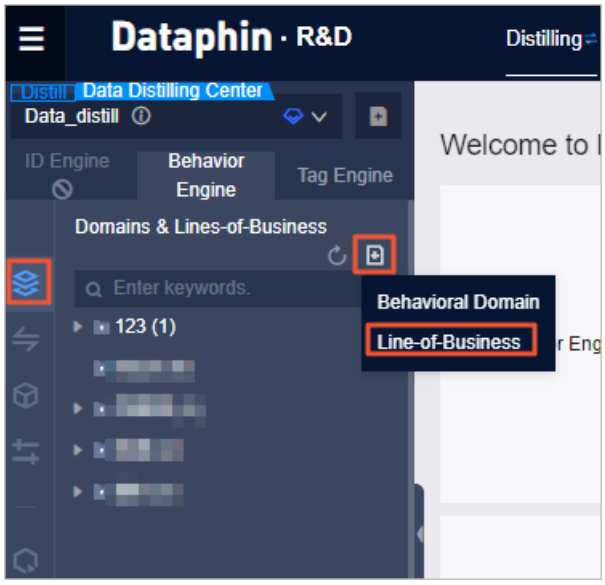
This topic describes how to create, edit, and delete a line-of-business.

Create a line-of-business

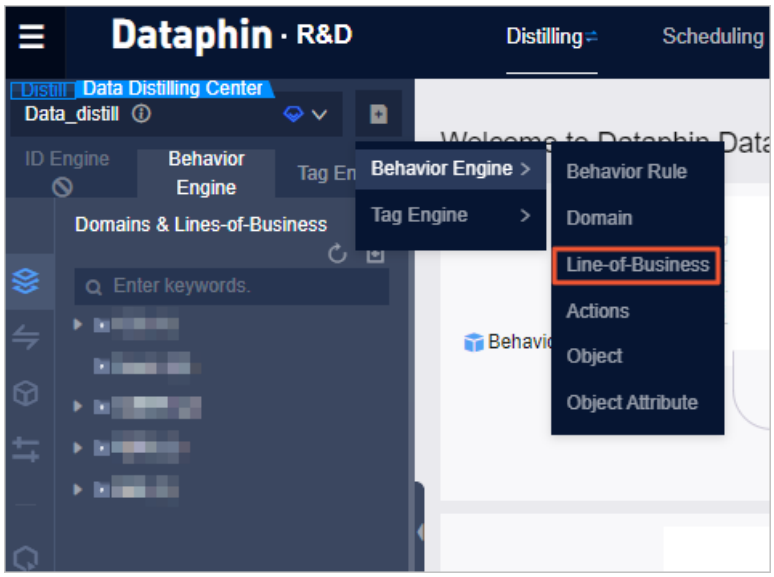
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



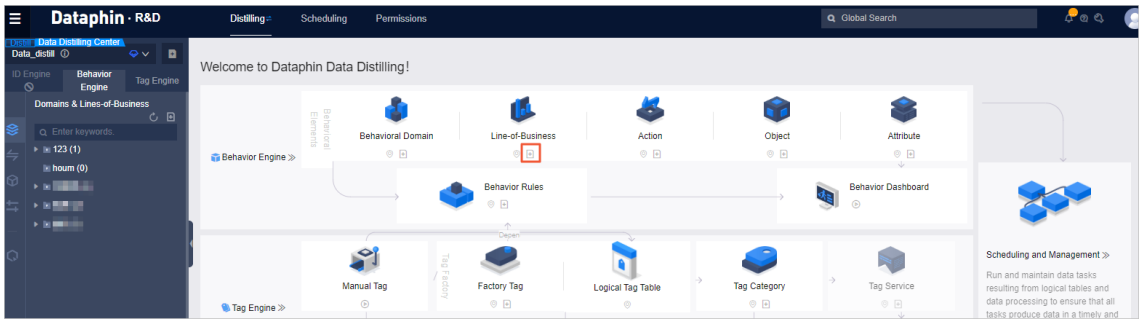
4. Open the **Create Line-of-Business** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon and select **Line-of-Business**.



- On the Distilling tab, click the icon next to the project name and choose **Behavior Engine > Line-of-Business.**

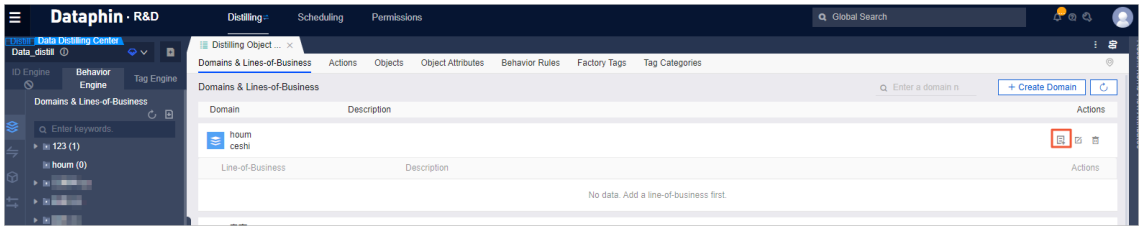


- On the Distilling tab, click the icon below **Line-of-Business** in the workspace.

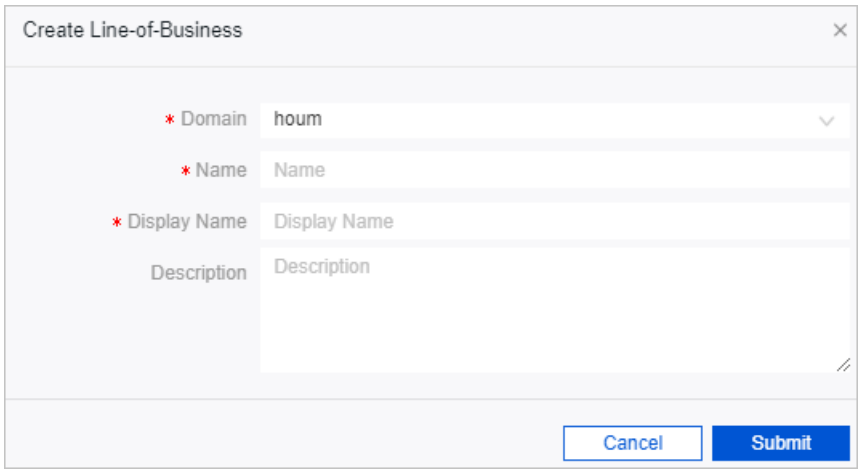


- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the

Domains & Lines-of-Business page that appears in the workspace, click the icon in the **Actions** column of the target behavioral domain.

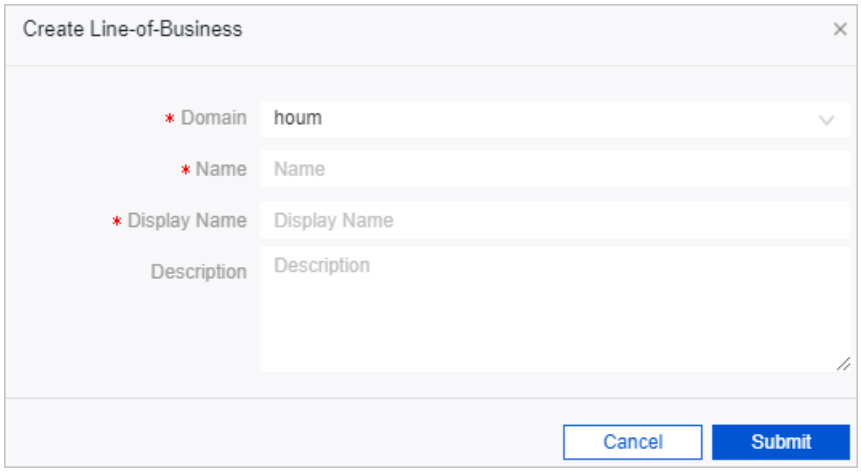


5. In the Create Line-of-Business dialog box that appears, set the **Domain**, **Name**, **Display Name**, and **Description** parameters. Then, click **Submit**.



> **Note**
>
> - The name and display name of the line-of-business cannot be the same as those of the existing lines-of-business.
> - A line-of-business depends on a behavioral domain. When you create a line-of-business, select the behavioral domain where the line-of-business resides from the drop-down list.

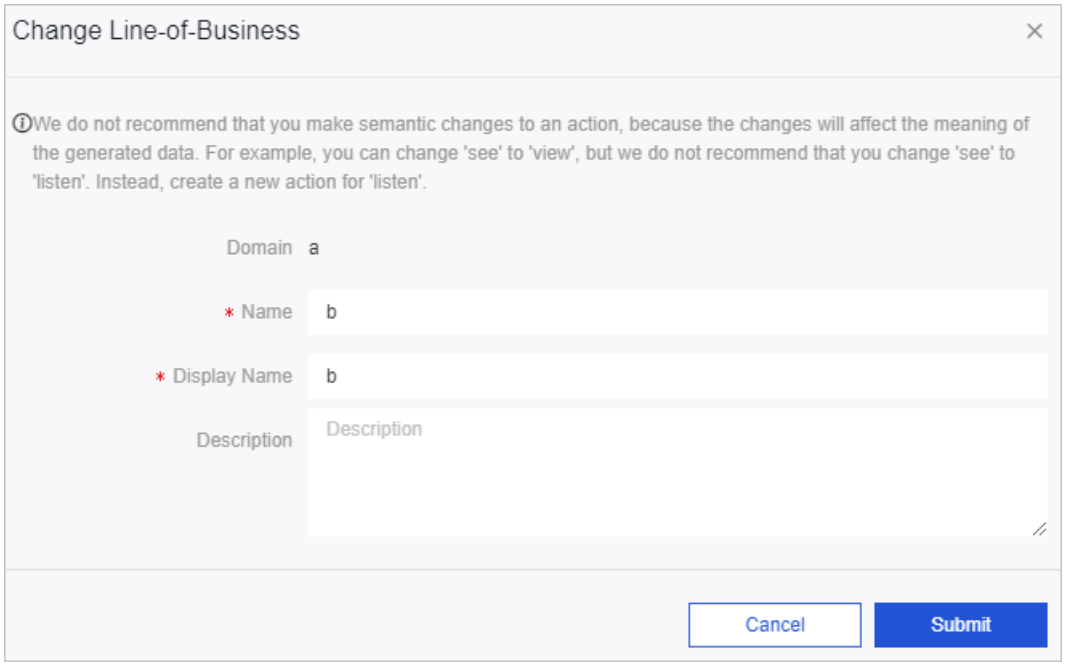
6. In the **Description** dialog box that appears, enter the comments on the line-of-business to be created. Then, click **OK**.



> **Note** A behavioral element is published to the production environment after it is created. Therefore, you must enter and submit comments when creating a behavioral element.

Edit a line-of-business

1. Open the **Change Line-of-Business** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side line-of-business list, click the target line-of-business. On the View Attributes tab that appears, click **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side line-of-business list, move the pointer over the icon next to the target line-of-business and select **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the **Domains & Lines-of-Business** page that appears in the workspace, click the icon in the **Actions** column of the target line-of-business.
2. In the Change Line-of-Business dialog box that appears, set the **Name**, **Display Name**, and **Description** parameters. Then, click **Submit**.



> **Note** The name and display name of the line-of-business cannot be the same as those of the existing lines-of-business.

3. In the **Description** dialog box that appears, enter the comments on the line-of-business to be modified. Then, click **OK**.

> Note A behavioral element is published to the production environment after it is edited. Therefore, you must enter and submit comments when editing a behavioral element.

Delete a line-of-business

1. Open the **Delete Line-of-Business** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side line-of-business list, click the target line-of-business. On the View Attributes tab that appears, click the icon and select **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side line-of-business list, move the pointer over the icon next to the target line-of-business and select **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Domains & Lines-of-Business on the left-side navigation submenu. Then, click **Domains & Lines-of-Business Object List** at the bottom of the left-side navigation pane. On the **Domains & Lines-of-Business** page that appears in the workspace, click the icon in the **Actions** column of the target line-of-business.
2. In the **Delete Line-of-Business** dialog box that appears, enter the comments on the line-of-business to be deleted. Then, click **OK**.

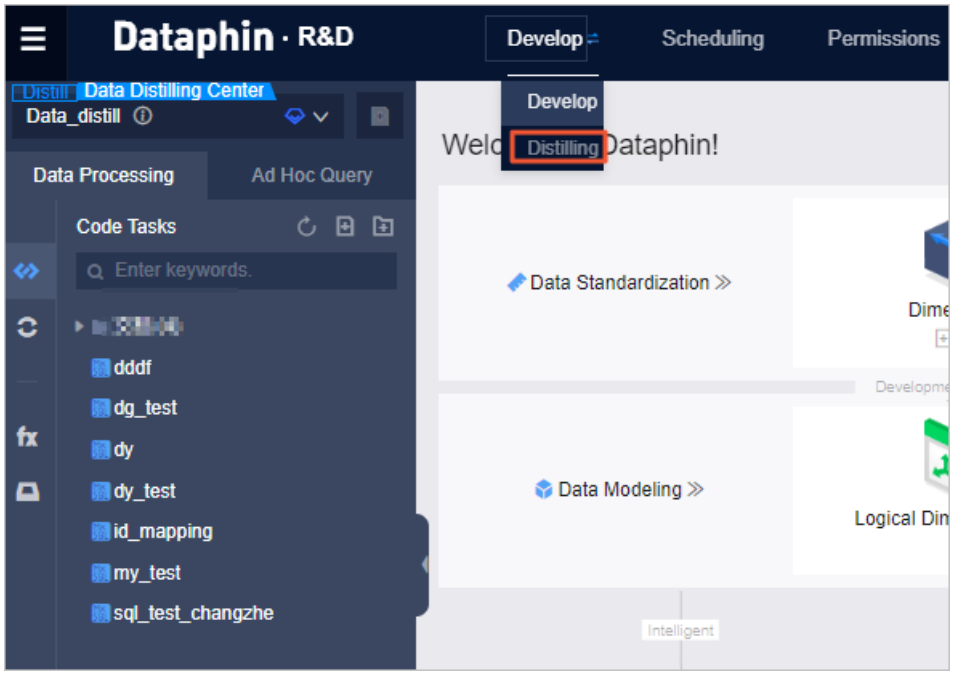
> Note
> - After a behavioral element is deleted, the relevant data is cleared from the production environment. Therefore, you must enter and submit comments when deleting a behavioral element.
> - You cannot delete a behavioral element on which a behavior rule depends.

9.10.2.4. Manage actions

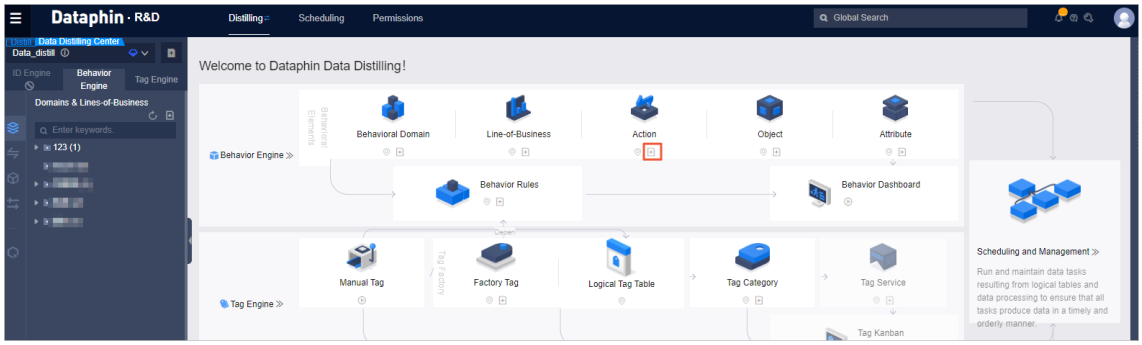
This topic describes how to create, edit, and delete an action.

Create an action

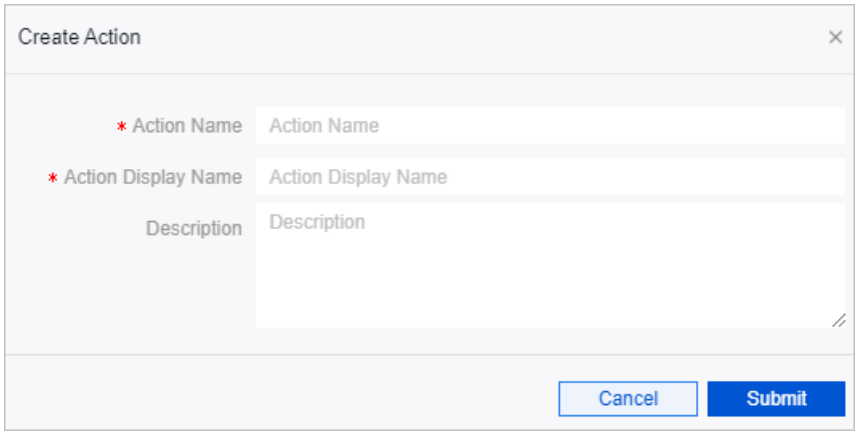
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



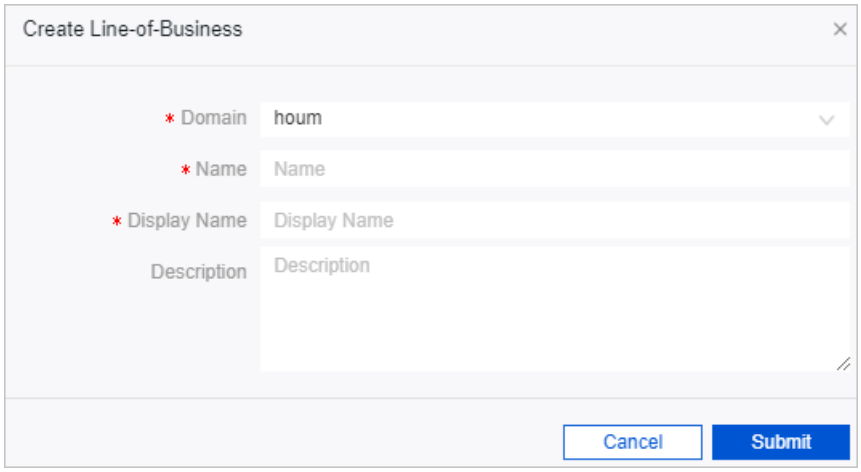
4. Open the **Create Action** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon.
   - On the Distilling tab, click the icon next to the project name and choose **Behavior Engine > Actions**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Actions on the left-side navigation submenu. Then, click **Actions Object List** at the bottom of the left-side navigation pane. On the **Actions** page that appears in the workspace, click **+ Create Action** in the upper-right corner.
   - On the Distilling tab, click the icon below **Action** in the workspace.



5. In the Create Action dialog box that appears, set the **Action Name**, **Action Display Name**, and **Description** parameters. Then, click **Submit**.



6. In the **Description** dialog box that appears, enter the comments on the action to be created. Then, click **OK**.

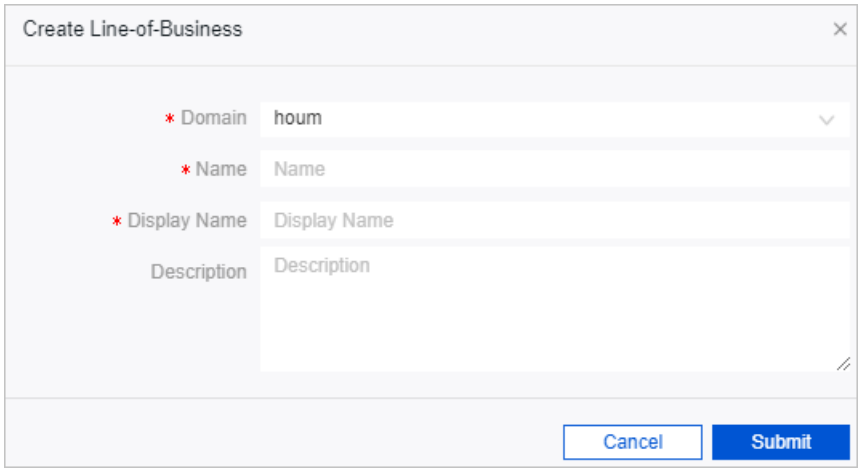


> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is created. Therefore, you must enter and submit comments when creating a behavioral element.

Edit an action

1. Open the **Change Action** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side action list, click the target action. On the View Attributes tab that appears, click **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side action list, move the pointer over the icon next to the target action and select **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Actions on the left-side navigation submenu. Then, click **Actions Object List** at the bottom of the left-side navigation pane. On the **Actions** page that appears in the workspace, click the icon in the **Actions** column of the target action.
2. In the Change Action dialog box that appears, set the **Action Name**, **Action Display Name**,

and **Description** parameters. Then, click **Submit**.



> **Note**
>
> - **Action Name**: the name of the action. The name can contain letters, digits, and underscores (_).
> - **Action Display Name**: the display name of the action. The display name can contain Chinese characters, digits, letters, underscores (_), and hyphens (-).
> - **Description**: the description of the action. The description can be up to 128 characters in length.

3. In the **Description** dialog box that appears, enter the comments on the action to be modified. Then, click **OK**.

> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is edited. Therefore, you must enter and submit comments when editing a behavioral element.

Delete an action

1. Open the **Delete Action** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side action list, click the target action. On the View Attributes tab that appears, click **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side action list, move the pointer over the icon next to the target action and select **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Actions on the left-side navigation submenu. Then, click **Actions Object List** at the bottom of the left-side navigation pane. On the **Actions** page that appears in the workspace, click the icon in the **Actions** column of the target action.
2. In the **Delete Action** dialog box that appears, enter the comments on the action to be deleted. Then, click **OK**.

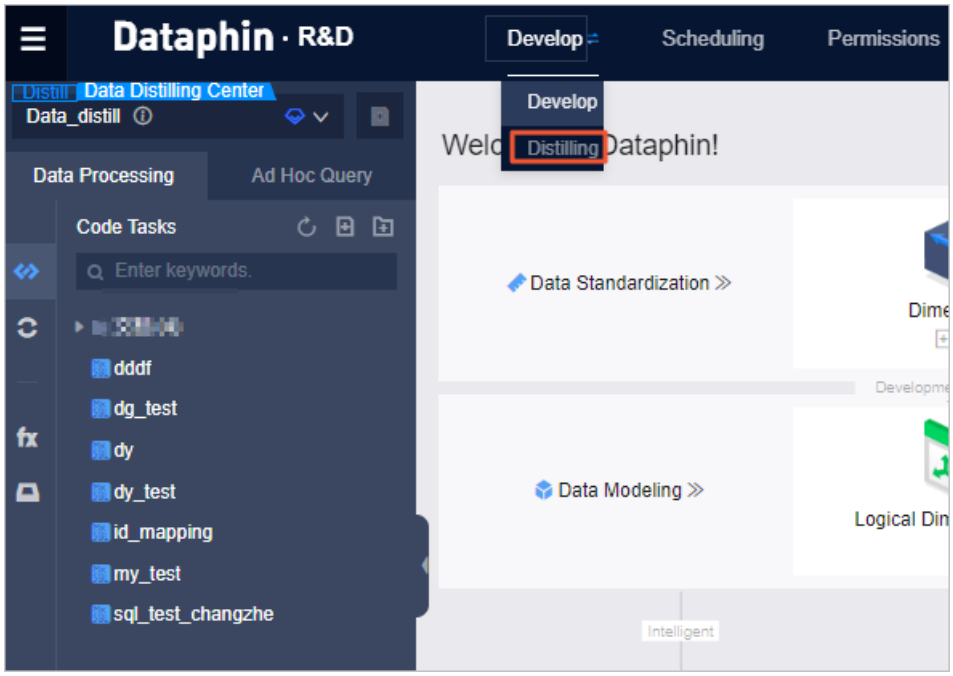
> **Note** A behavioral element is published to the production environment after it is created and is deleted from the production environment after the deletion operation. Therefore, you must enter and submit comments when deleting a behavioral element.

9.10.2.5. Manage objects

This topic describes how to create, edit, and delete an object.

Create an object

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



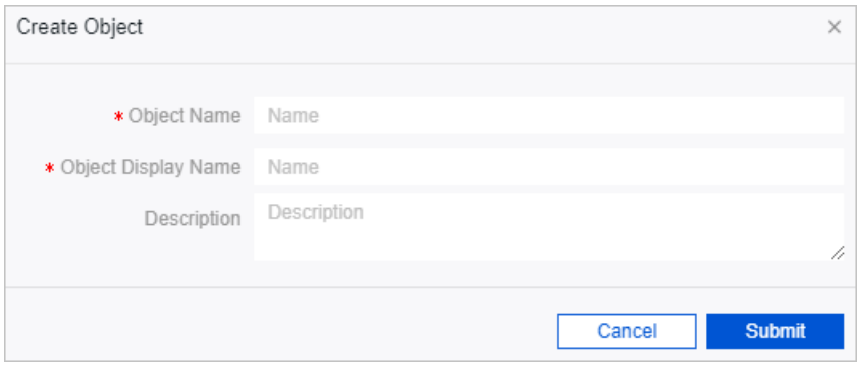
4. Open the Create Object dialog box in one of the following ways: On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon.

   On the Distilling tab, click the icon next to the project name and choose Behavior Engine > **Object**.

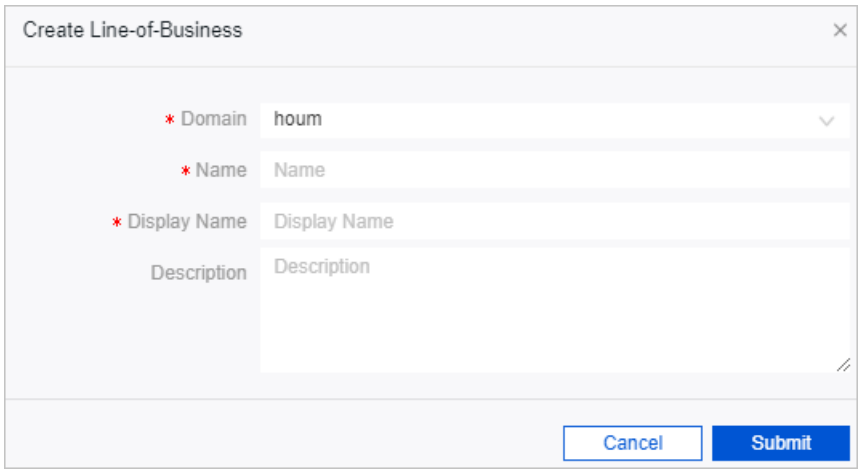
   On the Distilling tab, click the icon below **Object** in the workspace.

   On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Objects on the left-side navigation submenu. Then, click **Objects Object List** at the bottom of the left-side navigation pane. On the Objects page that appears in the workspace, click **+ Create Object** in the upper-right corner.
5. In the Create Object dialog box that appears, set the **Object Name**, **Object Display Name**,

and **Description** parameters. Then, click **Submit**.



6. In the **Description** dialog box that appears, enter the comments on the object to be created. Then, click **OK**.



> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is created. Therefore, you must enter and submit comments when creating a behavioral element.

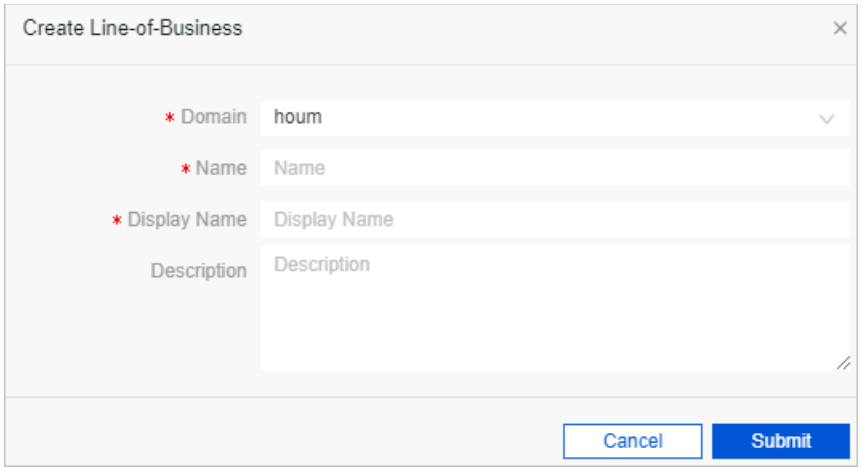
Edit an object

1. On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object list, click the target object. On the View Attributes tab that appears, click Change.

   To edit an object, you can move the pointer over the icon next to the target object in the left-side object list and select **Change**.

   You can also click Objects Object List at the bottom of the left-side navigation pane. On the Objects page that appears in the workspace, click the icon in the **Actions** column of the target object.

2. In the **Change Object** dialog box that appears, set the **Object Name**, **Object Display Name**, and **Description** parameters. Then, click **Submit**.



3. In the **Description** dialog box that appears, enter the comments on the object to be modified. Then, click **OK**.

> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is edited. Therefore, you must enter and submit comments when editing a behavioral element.

Delete an object

1. On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object list, click the target object. On the View Attributes tab that appears, click **Delete**.

   To delete an object, you can move the pointer over the icon next to the target object in the left-side object list and select **Delete**.

   You can also click Objects Object List at the bottom of the left-side navigation pane. On the Objects page that appears in the workspace, click the icon in the **Actions** column of the target object.

2. In the **Delete Object** dialog box that appears, enter the comments on the object to be deleted. Then, click **OK**.

> **Note** A behavioral element is published to the production environment after it is created and is deleted from the production environment after the deletion operation. Therefore, you must enter and submit comments when deleting a behavioral element.

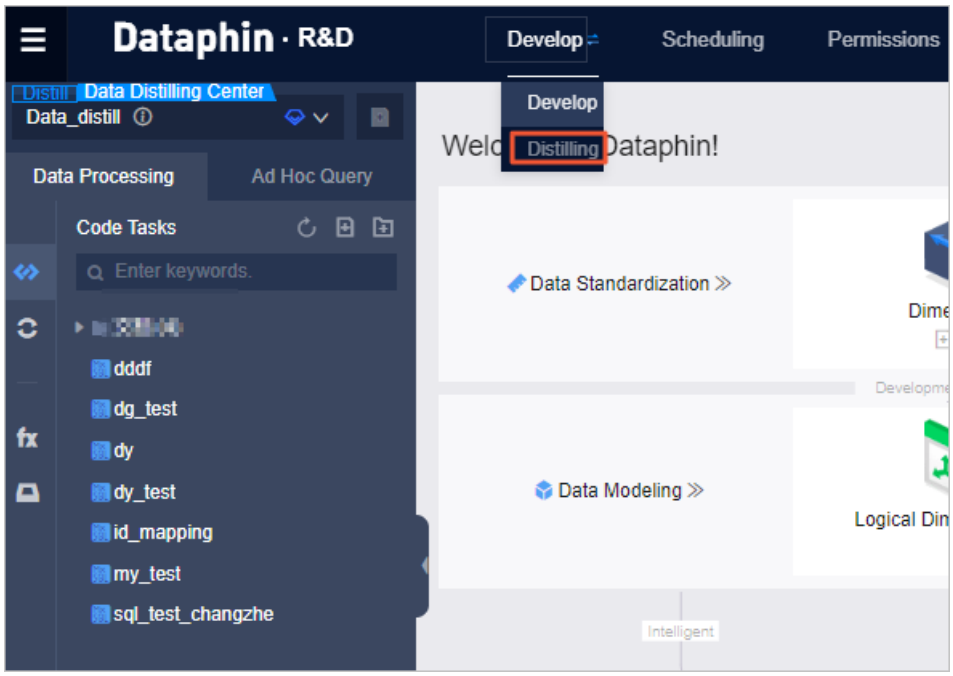
9.10.2.6. Manage object attributes

This topic describes how to create, edit, and delete an object attribute.

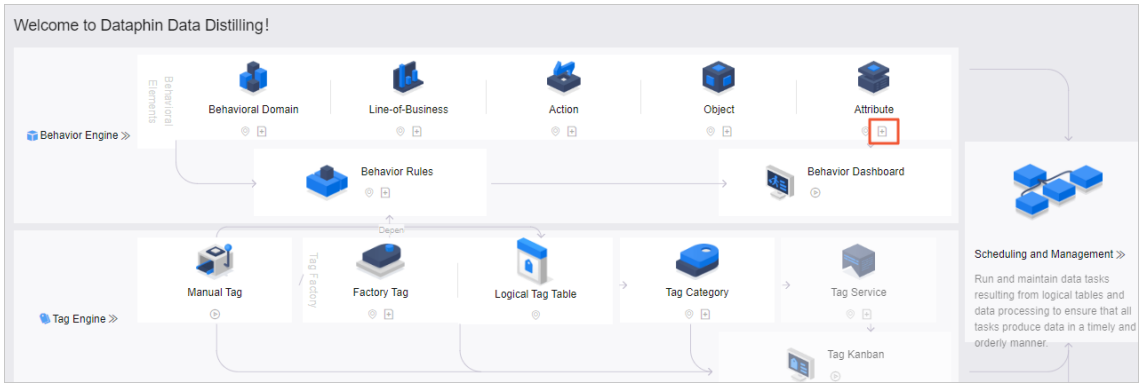
Create an object attribute

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and

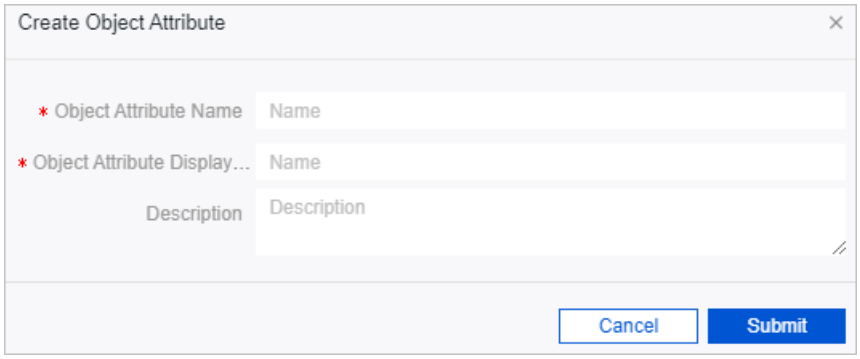
select **Distilling**.



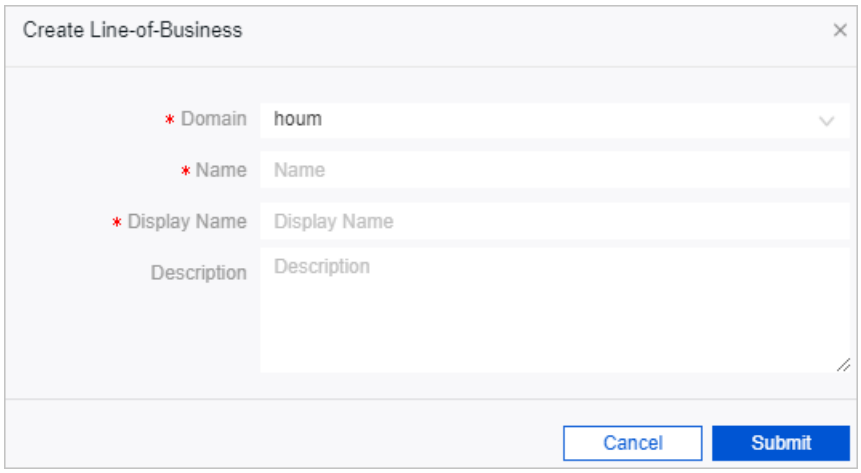
4. Open the **Create Object Attribute** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon.
   - On the Distilling tab, click the icon next to the project name and choose **Behavior Engine > Object Attribute**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Object Attributes on the left-side navigation submenu. Then, click **Object Attributes Object List** at the bottom of the left-side navigation pane. On the **Object Attributes** page that appears in the workspace, click **+ Create Object Attribute** in the upper-right corner.
   - On the Distilling tab, click the icon below **Attribute** in the workspace.



5. In the Create Object Attribute dialog box that appears, set the **Object Attribute Name**, **Object Attribute Display Name**, and **Description** parameters. Then, click **Submit**.



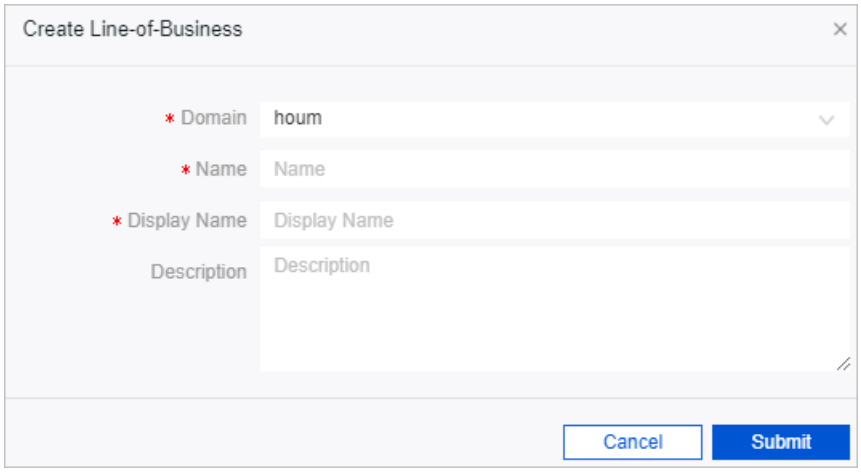
6. In the **Description** dialog box that appears, enter the comments on the object attribute to be created. Then, click **OK**.



> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is created. Therefore, you must enter and submit comments when creating a behavioral element.

Edit an object attribute

1. Open the **Change Object Attribute** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object attribute list, click the target object attribute. On the View Attributes tab that appears, click **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object attribute list, move the pointer over the icon next to the target object attribute and select **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Object Attributes on the left-side navigation submenu. Then, click **Object Attributes Object List** at the bottom of the left-side navigation pane. On the **Object Attributes** page that appears in the workspace, click the icon in the **Actions** column of the target object attribute.
2. In the Change Object Attribute dialog box that appears, set the **Object Attribute Name**, **Object Attribute Display Name**, and **Description** parameters. Then, click **Submit**.



3. In the **Description** dialog box that appears, enter the comments on the object attribute to be modified. Then, click **OK**.

> **Note** Behavioral elements have no states. A behavioral element is published to the production environment after it is edited. Therefore, you must enter and submit comments when editing a behavioral element.

Delete an object attribute

1. Open the **Delete Object Attributes** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object attribute list, click the target object attribute. On the View Attributes tab that appears, click **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side object attribute list, move the pointer over the icon next to the target object attribute and select **Delete**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Object Attributes on the left-side navigation submenu. Then, click **Object Attributes Object List** at the bottom of the left-side navigation pane. On the **Object Attributes** tab that appears in the workspace, click the icon in the **Actions** column of the target object attribute.
2. In the **Delete Object Attributes** dialog box that appears, enter the comments on the object attribute to be deleted. Then, click **OK**.

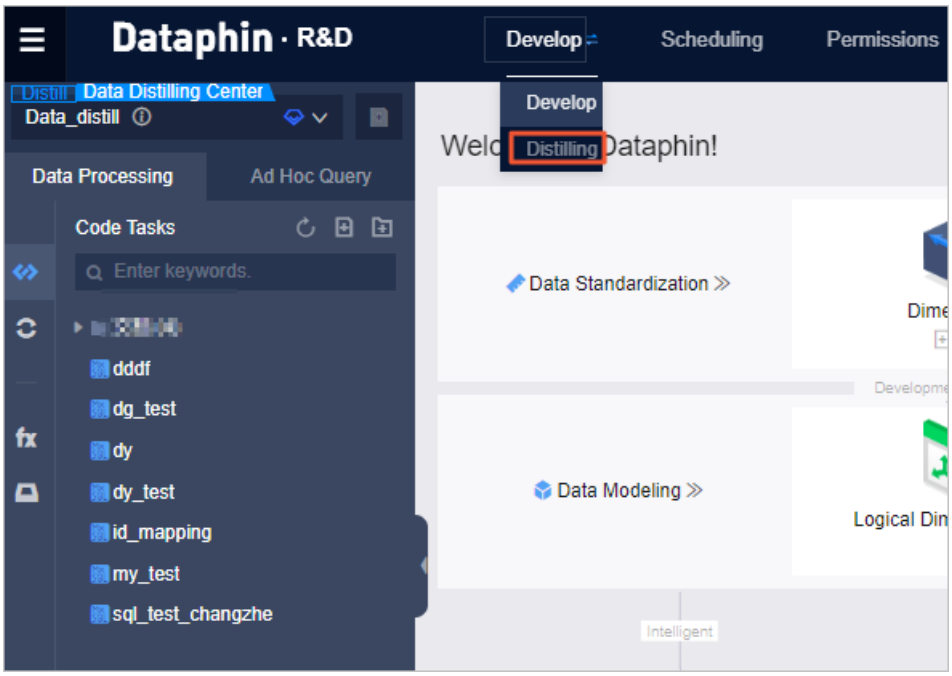
> **Note** A behavioral element is published to the production environment after it is created and is deleted from the production environment after the deletion operation. Therefore, you must enter and submit comments when deleting a behavioral element.

9.10.2.7. Create a behavior rule

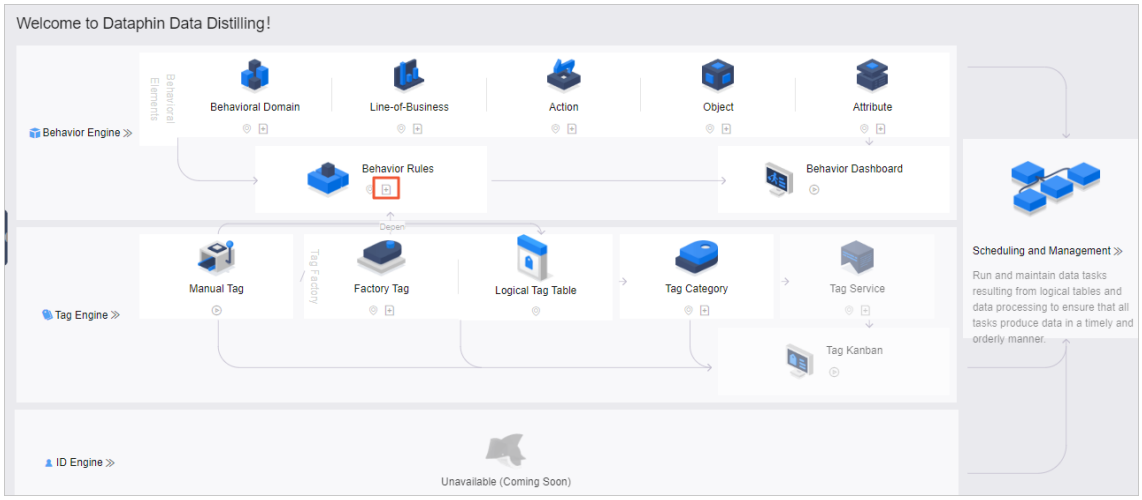
This topic describes how to create a behavior rule and configure the scheduling policy and conversion task settings for the rule.

1. Log on to the Dataphin console.

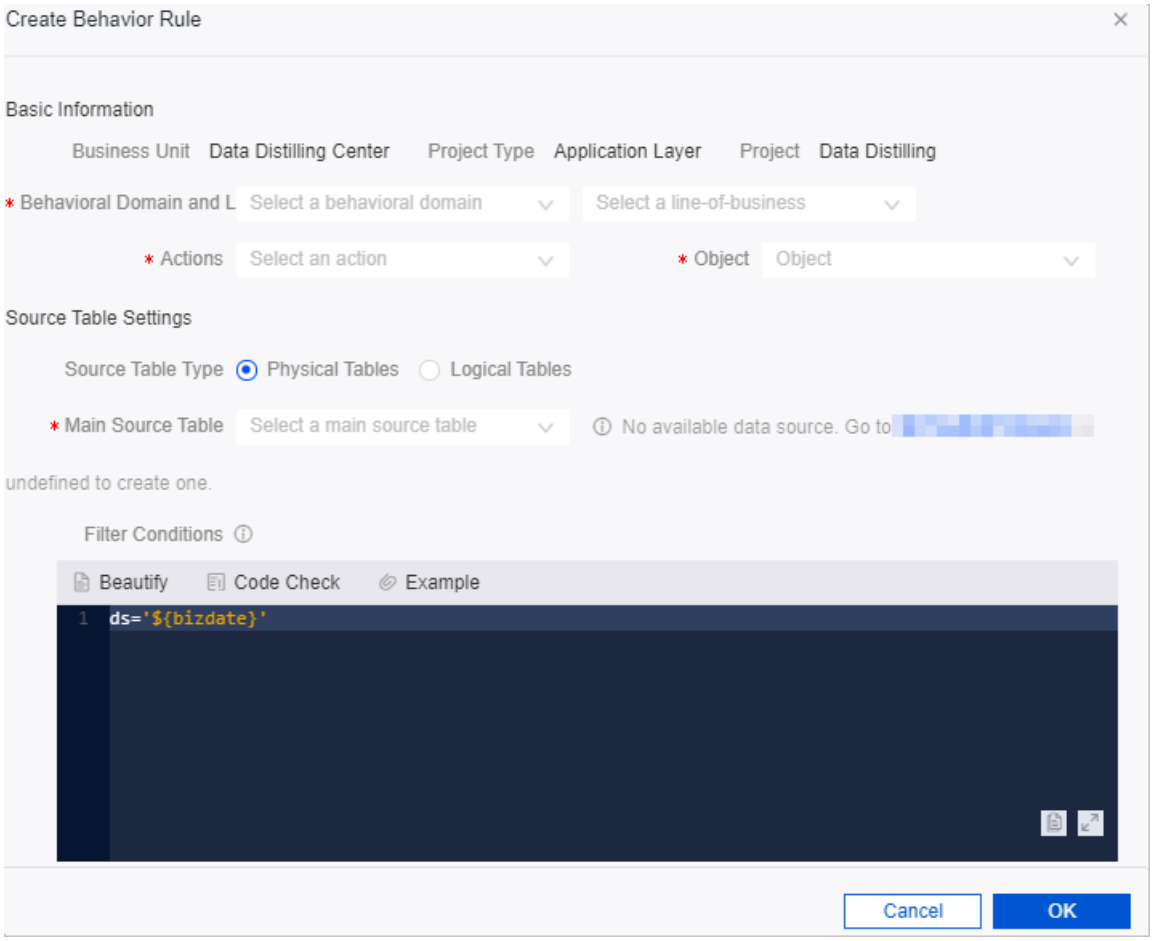
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



4. Open the **Create Behavior Rule** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon.
   - On the Distilling tab, click the icon next to the project name and choose **Behavior Engine > Behavior Rule**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click **Behavior Rules Object List** at the bottom of the left-side navigation pane. On the **Behavior Rules** page that appears in the workspace, click **+ Create Behavior Rule** in the upper-right corner.
   - On the Distilling tab, click the icon below **Behavior Rules** in the workspace.



5. In the **Create Behavior Rule** dialog box that appears, set the parameters as prompted.



Section	Parameter	Description
Basic Information	Behavioral Domain and Line-of-Business	Select a behavioral domain and a line-of-business.
	Actions	Select an action.
	Object	Select an object.

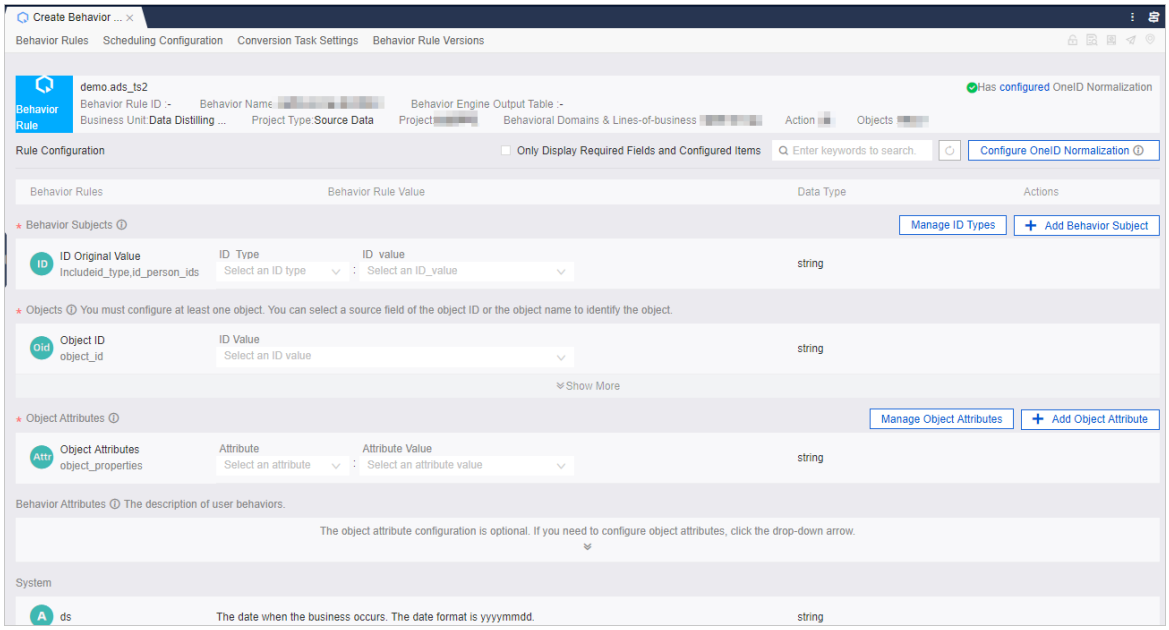
Section	Parameter	Description
Source Table Settings	Source Table Type	Select Physical Tables or Logical Tables.
	Main Source Table	○ If you set Source Table Type to Physical Tables, you can only select the main source table. ○ If you set Source Table Type to Logical Tables, select a business unit and then select the main source table.
	Filter Conditions	Enter filter conditions, such as `ds=${bizdate}`, used to cleanse data records in the source table.

> **Note**
> - If you select a main source table in the development environment, the system cannot automatically parse the task dependencies during scheduling configuration.
> - You must focus on the source table of the behavioral data and the filter conditions to make sure that the source table meets the behavioral data requirements. Otherwise, you cannot create a behavior rule.

6. Click **OK** to go to the **Rule Configuration** page.

   On the **Rule Configuration** page that appears, you can view the basic information and source table settings that have been configured for the behavior rule. To modify the basic information or source table settings of the behavior rule, click **Behavior Rules** at the top of the page. In the **Behavior Rules** dialog box that appears, set the parameters.

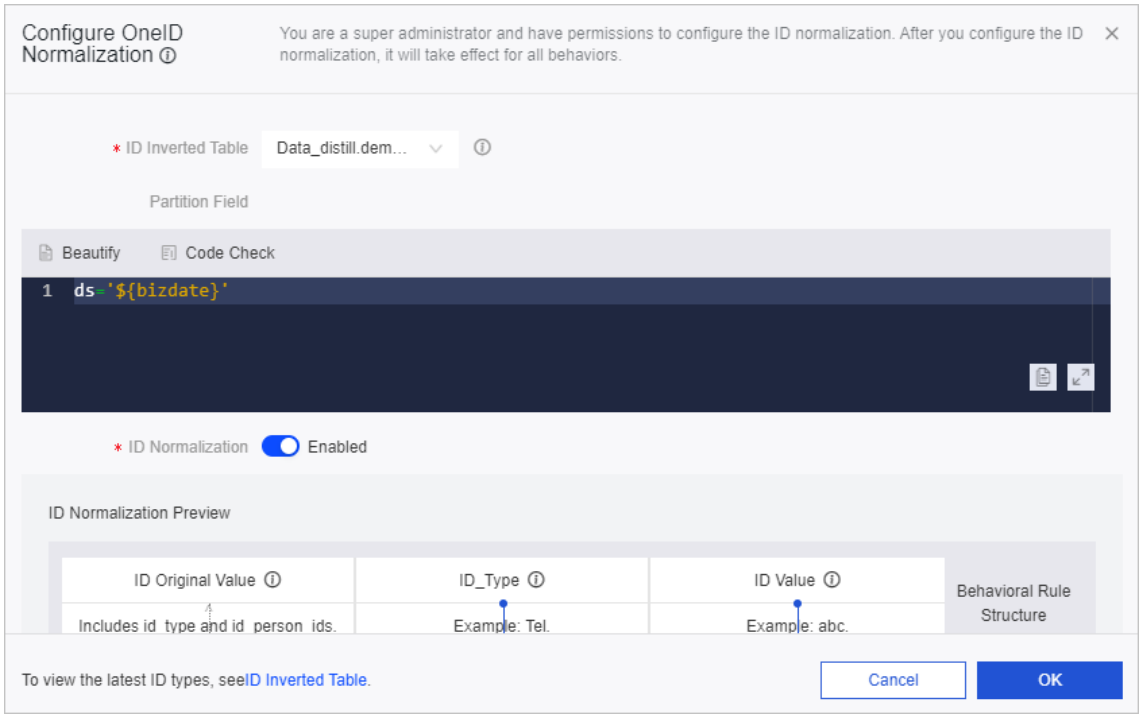
7. On the Rule Configuration page, set the parameters as prompted.



Section	Parameter	Description
Behavior Subjects	**ID Original Value**	Select an ID type from the **ID_Type** drop-down list and select an ID value from the **ID_value** drop-down list. ○ You can click **Manage ID Types**. In the Manage ID Types dialog box that appears, click **+ Add ID Type** to add a custom ID type. The built-in ID types include **OneID**, **Mobile**, **UserID**, **Email**, and **IPAddress**. ○ To add one or more behavior subjects, click **+ Add Behavior Subject**. You can delete an existing behavior subject only when more than one behavior subject exists. **Note** You can delete a custom ID type, but cannot delete or modify the built-in ID types.
Objects	**Object ID**	Select an ID value from the **ID Value** drop-down list.
	Object Display Name	Select a display name for the object from the **Object Display Name** drop-down list. **Note** In the Objects section, the object ID and display name are only used to identify objects and cannot be used as object attributes. If you want to use the object ID or display name as an object attribute, add the object ID or display name to the **Object Attributes** section.

Section	Parameter	Description
Object Attributes	Object Attributes	An object attribute is the factual description of an object, such as the name, year, or director of a video object. Select an object attribute from the **Attribute** drop-down list and select an attribute value from the **Attribute Value** drop-down list. ◦ If no object attribute is available, you can click **Manage Object Attributes** to go to the **Object Attributes** page. On the Object Attributes page that appears, you can create object attributes and manage existing object attributes. For more information, see Manage object attributes. ◦ You can click **+ Add Object Attribute** to add one or more object attributes. You can also delete existing object attributes. However, you must retain at least one object attribute. **Note** Make sure that the source fields of the attribute values are stable to prevent the subsequent tag calculation results from being affected by the attribute value changes. Generally, each attribute value of an object is stable and unique. You can select an object attribute that has been defined as a behavioral element from the Attribute drop-down list.
Behavior Attributes	Times Occurred	Optional. If the source table you select contains a field that indicates the number of occurrences of the behavior, you can click the icon and select an attribute value from the **Attribute Value** drop-down list to add the field as a behavior attribute.
System	ds	The partition field that is automatically created based on the system data timestamp for the Behavior Engine output table, in the format of yyyymmdd.

◦ If you have a standard ID inverted table, click **Configure OneID Normalization** to import the ID inverted table. The enabled ID normalization feature takes effect for all rules.

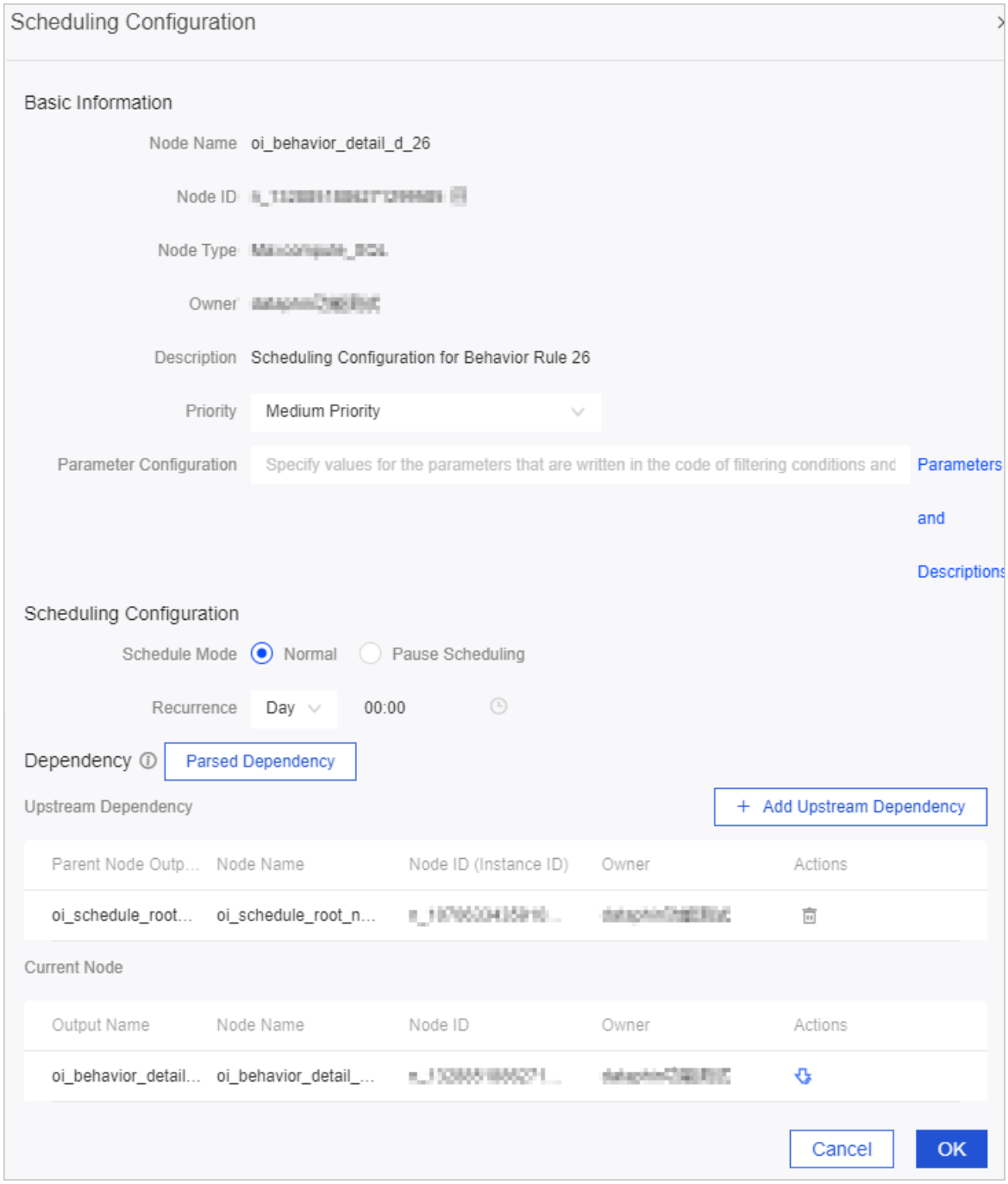


You can click ID Inverted Table at the bottom of the dialog box to view the table schema. If you have selected the ID type, you can click ID Type to view the details of the inverted table.

> **Note**
>
> - Only the super administrator can enable or disable ID normalization.
> - Based on the OneID methodology, the ID inverted table uses algorithms to connect different IDs of the same user through OneID.
> - After ID normalization is enabled, if the selected ID type of a behavior subject is normalized and contained in the ID inverted table, the result table generates records based on OneID and the ID type. If the ID type of the behavior subject is not normalized or is not contained in the ID inverted table, the result table generates records based on the ID type.

- **If you do not have a standard ID inverted table, you can also choose Develop > Data Processing in the Data_distill project to create a task and import the ID inverted table. Make sure that the output name of the task node for the ID inverted table is in the format of Project name.Table name, for example,** `Data_distill.Inverted table name` **.**

8. **On the Rule Configuration page, click Scheduling Configuration at the top. In the Scheduling Configuration dialog box that appears, set the parameters as prompted. Then, click OK.**

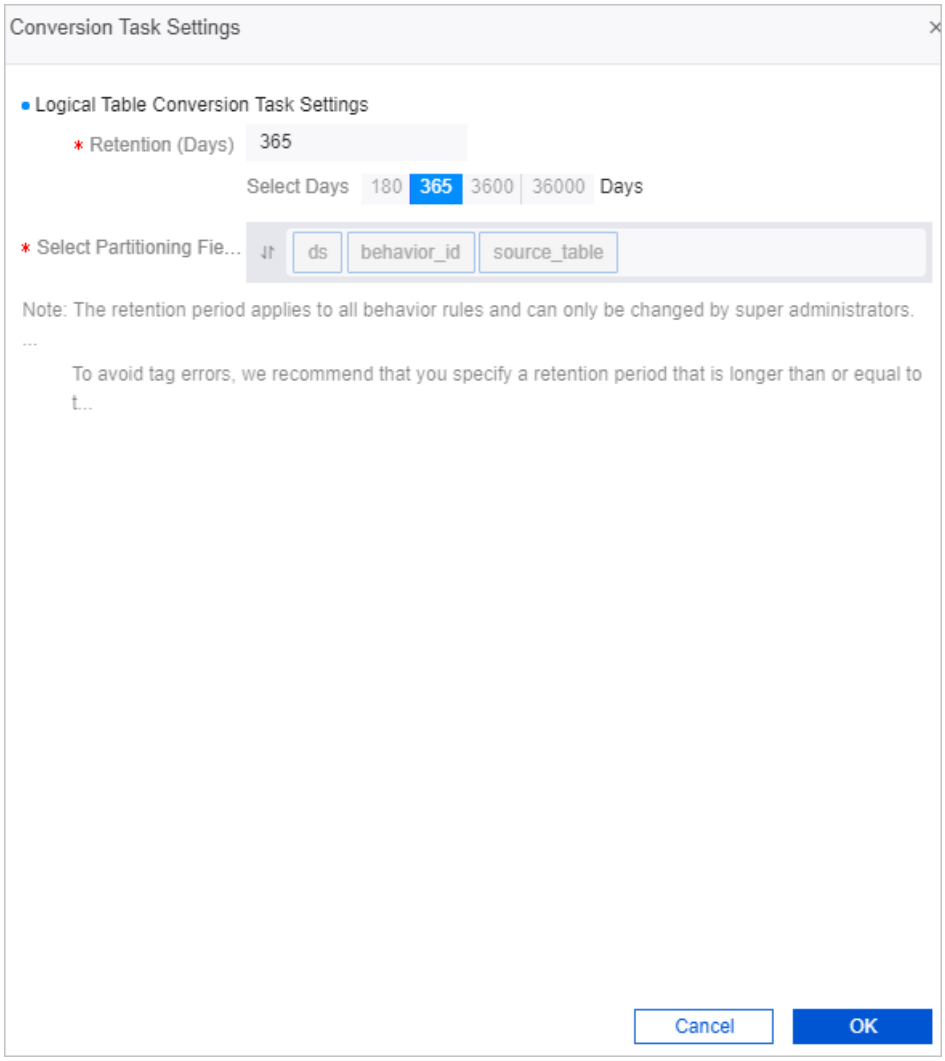


Section	Parameter	Description

Section	Parameter	Description
Basic Information	Priority	Dataphin automatically generates the node name, node ID, node type, owner, and description. You cannot modify these parameters. You can select the priority of the task corresponding to the behavior rule. Valid values: ○ Lowest Priority ○ Low Priority ○ Medium Priority ○ High Priority ○ Highest Priority
	Parameter Configuration	You can set variables for the filter conditions of the source table. For more information, see the page that appears after you click Parameters and Descriptions.
Scheduling Configuration	Schedule Mode	You can set Schedule Mode to Normal or Pause Scheduling.
	Recurrence	You can set Recurrence to Day, Week, or Month. ○ If you select Day, click the icon to specify the time of a day. ○ If you select Month, select a date from the drop-down list and then click the icon to specify the time of a day. ○ If you select Week, select a day in a week from the drop-down list and then click the icon to specify the time of a day.

Section	Parameter	Description
Dependency	Upstream Dependency	◦ You can click **Parsed Dependency**. Dataphin automatically parses the upstream dependency based on the selected source table. ◦ You can click **+ Add Upstream Dependency** to add nodes in the production environment that are from projects in Basic mode or Prod mode as the upstream dependency. **Note** ◦ If the source table is a logical table, the parent node parsed by the system cannot be deleted. ◦ If the source table is a physical table, the parent node parsed by the system can be deleted. ◦ If you change the source table, the parsed upstream nodes are cleared. You may need to manually parse the upstream dependency or modify the upstream nodes. ◦ If the automatic parsing fails, you can click **Parsed Dependency** to parse the dependency again. The parent node that you manually added will not be overwritten by the parsed parent node.
	Current Node	You can click the icon to view downstream nodes.

9. On the Rule Configuration page, click **Conversion Task Settings** at the top. In the Conversion Task Settings dialog box that appears, set the parameters as prompted. Then, click **OK**.



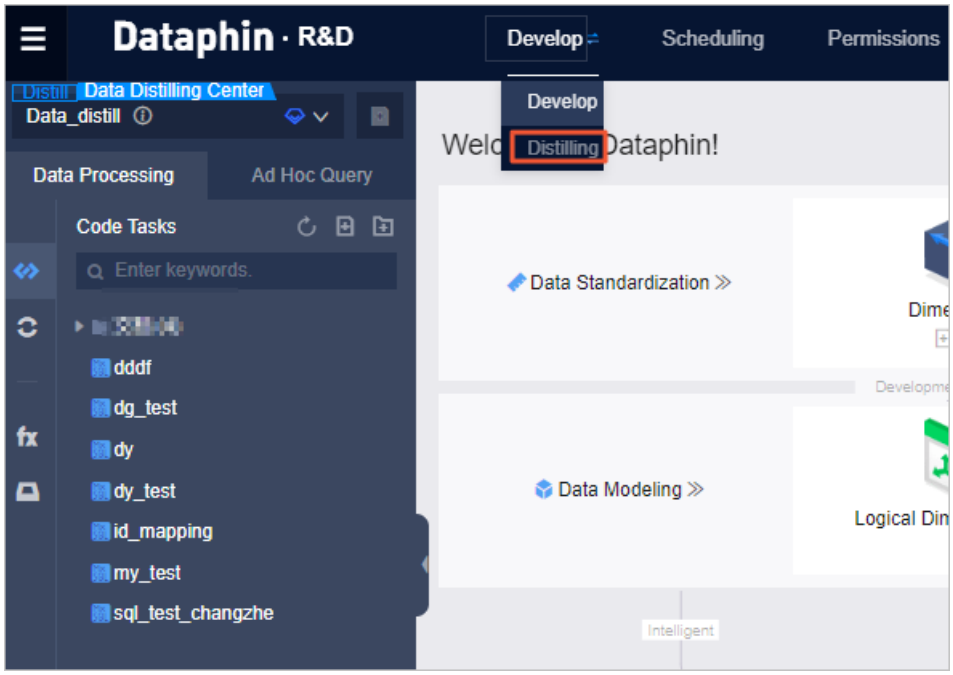
10. In the upper-right corner of the page, click the icon to preview the behavior rule, click the icon to save the configuration, or click the icon to submit the behavior rule.

9.10.2.8. Manage behavior rules

This topic describes how to manage behavior rules, for example, how to view and edit a behavior rule.

Edit a behavior rule

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



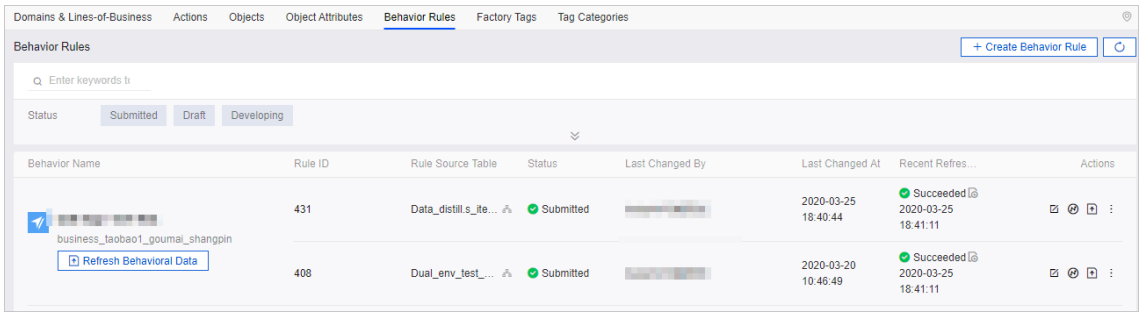
4. Go to the **Rule Configuration** page in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click **Change**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click **Behavior Rules Object List** at the bottom of the left-side navigation pane. On the **Behavior Rules** page that appears in the workspace, click the icon in the **Actions** column of the target behavior rule.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the target behavior rule and select **Change**.
5. On the **Rule Configuration** page that appears, click **Behavior Rules** at the top. In the Behavior Rules dialog box that appears, you can modify the basic information and source table settings of the behavior rule. You can also modify the rule configuration information on the Rule Configuration page. For more information, see Create a behavior rule.
6. In the upper-right corner of the page, click the icon to preview the behavior rule, click the icon to save the configuration, or click the icon to submit the edited behavior rule.
7. On the **Rule Configuration** page, click **Behavior Rule Versions** at the top. In the Node Version dialog box that appears, you can view the earlier versions of the submitted behavior rule.

View a behavior rule

1. View the information about a behavior rule in different dimensions in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, click the

target behavior rule. On the View Attributes tab that appears, you can view the information about the behavior rule in the **Basic Information**, **Consumption Information**, **Behavior Subjects**, **Objects**, **Object Attributes**, and **Behavior Attributes** sections.

- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click**Behavior Rules Object List** at the bottom of the left-side navigation pane. On the Behavior Rules page that appears in the workspace, set the filter conditions to view multiple behavior rules. You can click the ID in the**Rule ID** column of a behavior rule to view the information about the behavior rule on the View Attributes tab that appears.



  - **Behavior Name**: the name of the behavior.
  - **Rule ID**: the ID of the behavior rule corresponding to the behavior.
  - **Rule Source Table**: the source table of the behavior rule.
  - **Status**: the development status of the behavior rule. The valid values are Draft, Submitted, and Developing. **Developing** indicates that the submitted behavior rule is edited and saved but not submitted again, and the behavior rule is involved in task scheduling in the production environment based on the previous configuration. **Submitted** indicates that the behavior rule is involved in task scheduling in the production environment. **Draft** indicates that the new behavior rule is not submitted.
  - **Recent Refresh Status**: the status of retroactive data generation for the behavior rule. The valid values include No Retroactive Data, Running, Succeeded, and Failed. You can click the Refresh Log icon next to the status of retroactive data generation for a behavior rule to view all the retroactive data generation records of the behavior rule. In the **Refresh Records** window that appears, click the icon in the **Refresh Status** column of a retroactive data generation record. On the **Scheduling** tab that appears, you can view the directed acyclic graph (DAG) of the retroactive data generation instance.

Generate retroactive data

1. Open the**Generate Retroactive Data for Behavior Rules** dialog box in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the behavior to which the target behavior rule belongs and select**Refresh**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the

pointer over the icon next to the target behavior rule and select**Refresh**.

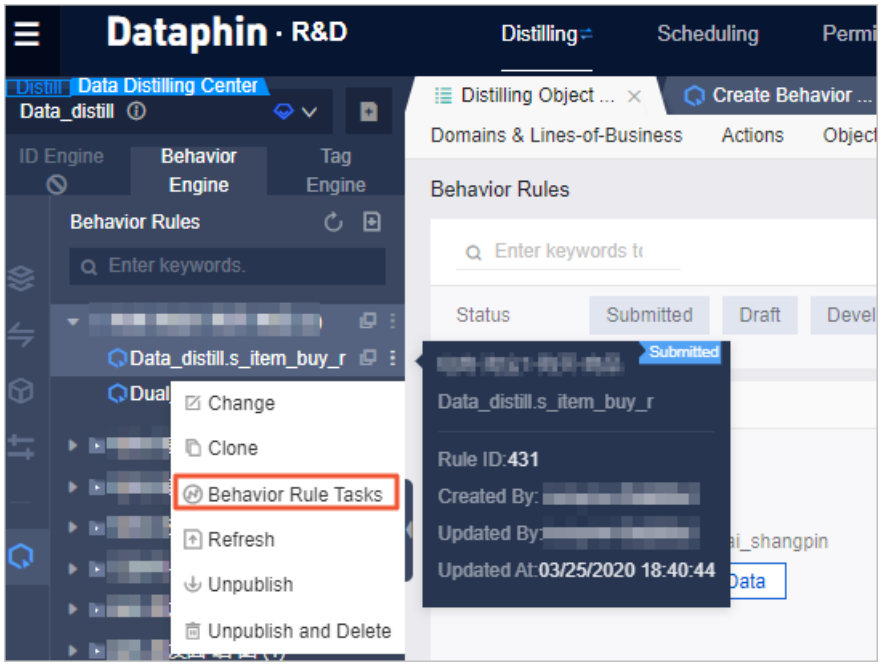
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click**Generate Retroactive Data**.
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click**Behavior Rules Object List** at the bottom of the left-side navigation pane. On the**Behavior Rules** page that appears in the workspace, click the icon in the Actions column of the target behavior rule to open the**Generate Retroactive Data for Behavior Rules** dialog box.

2. In the**Generate Retroactive Data for Behavior Rules** dialog box that appears, set**Effective Period** and**Instance Name** as prompted. Then, click**OK**.

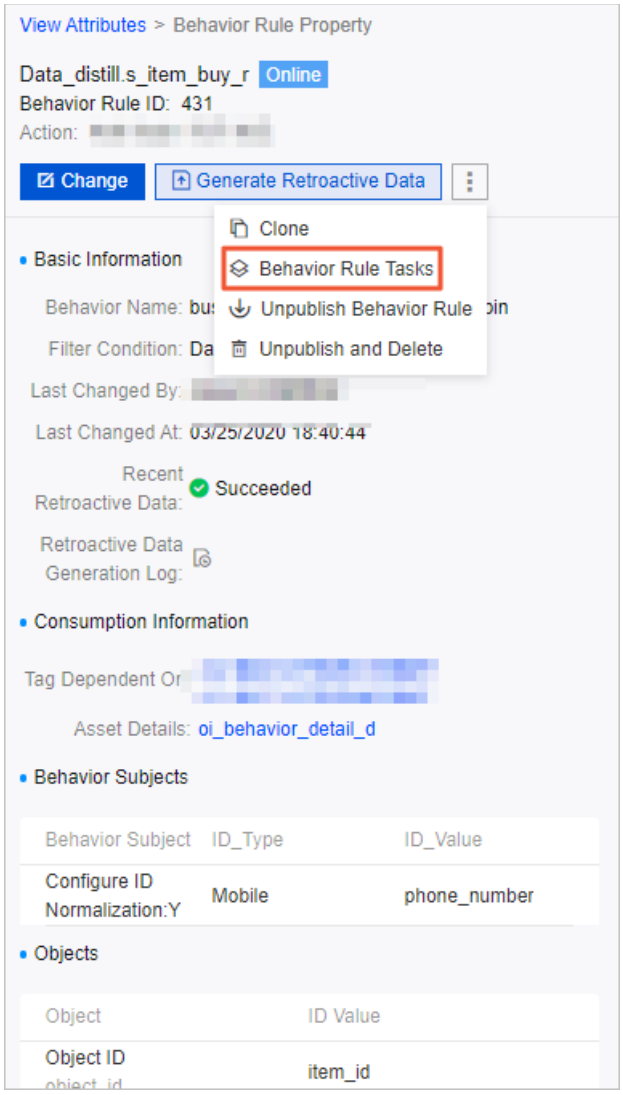
> **Note** You can only generate retroactive data for behavior rules in the Submitted or Developing state.

View a behavior rule task

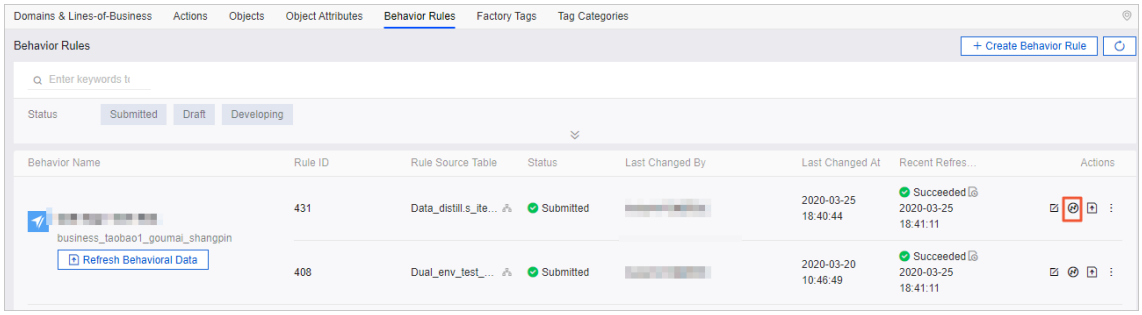
1. Go to the**Behavior Rule Tasks** page of the Scheduling module in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the target behavior rule and select**Behavior Rule Tasks**.



   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click the icon and select**Behavior Rule Tasks**.



- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click**Behavior Rules Object List** at the bottom of the left-side navigation pane. On the**Behavior Rules** page that appears in the workspace, click the icon in the Actions column of the target behavior rule.



2. On the**Behavior Rule Tasks** page of the Scheduling module that appears, view and maintain the production data.

> **Note** You can only view the behavior rule task for behavior rules in the Submitted or Developing state.

Clone a behavior rule

1. Go to the **Rule Configuration** page in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the target behavior rule and select **Clone**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click the icon and select **Clone**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click **Behavior Rules Object List** at the bottom of the left-side navigation pane. On the **Behavior Rules** page that appears in the workspace, click the icon in the Actions column of the target behavior rule and select **Clone**.
2. On the **Rule Configuration** page that appears, view the detailed configuration information about the cloned behavior rule. You can also modify the configuration based on your business requirements. For more information about how to modify the configuration, see Edit a behavior rule.

Unpublish a behavior rule

1. Open the **Tip** dialog box to unpublish a behavior rule in one of the following ways:
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the target behavior rule and select **Unpublish**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click the icon and select **Unpublish Behavior Rule**.
   - On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click **Behavior Rules Object List** at the bottom of the left-side navigation pane. On the **Behavior Rules** page that appears in the workspace, click the icon in the Actions column of the target behavior rule and select **Unpublish**.
2. In the **Tip** dialog box that appears, click **OK**. After a behavior rule is unpublished, it enters the Draft state.

> **Note** You cannot unpublish a behavior rule on which a tag in the Submitted or Developing state depends.

Unpublish and delete a behavior rule

1. Open the Tip dialog box to unpublish and delete a behavior rule in one of the following

ways:

- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side behavior rule list, move the pointer over the icon next to the target behavior rule and select **Unpublish and Delete**.
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. In the left-side behavior rule list, click the target behavior rule. On the View Attributes tab that appears, click the icon and select **Unpublish and Delete**.
- On the Distilling tab, click Behavior Engine in the left-side navigation pane and click Behavior Rules on the left-side navigation submenu. Then, click **Behavior Rules Object List** at the bottom of the left-side navigation pane. On the **Behavior Rules** page that appears in the workspace, click the icon in the Actions column of the target behavior rule and select **Unpublish and Delete**.

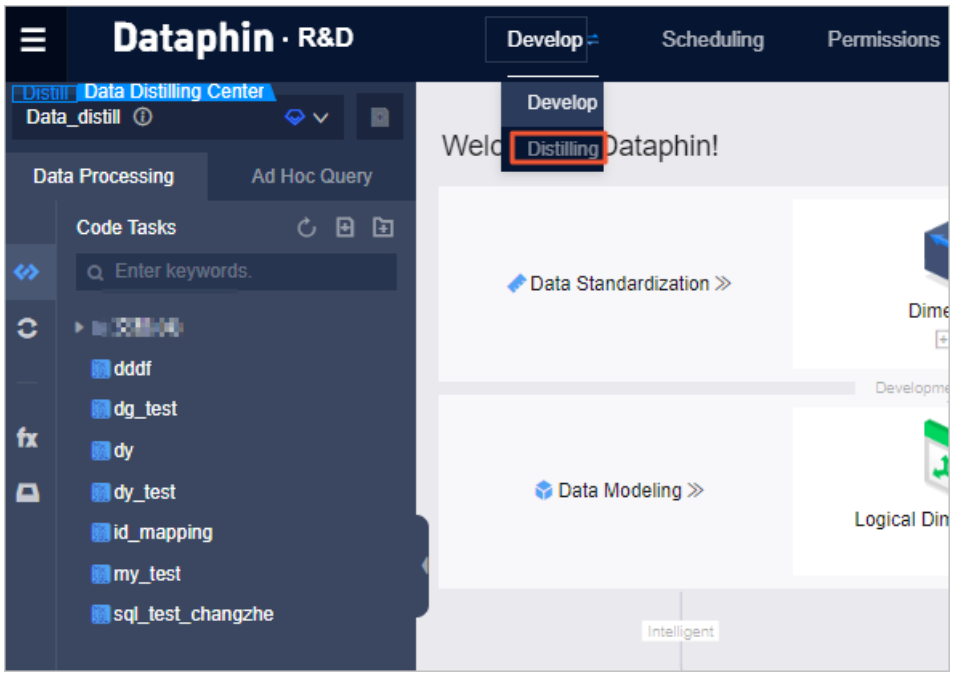
2. In the Tip dialog box that appears, enter the comments and click **OK**.

> **Note** You cannot unpublish and delete a behavior rule on which a tag in the Submitted or Developing state depends.

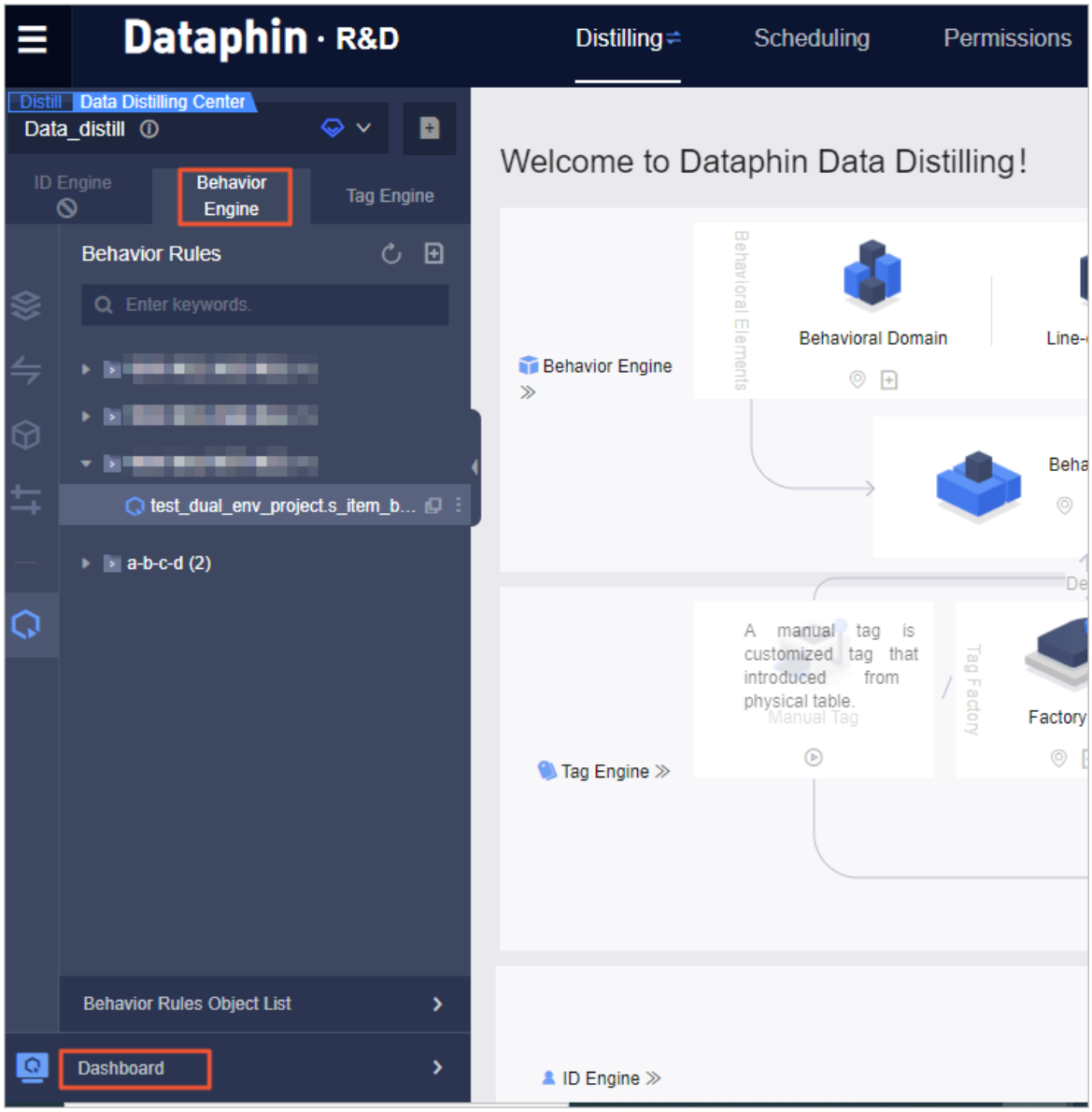
9.10.2.9. Dashboard

This topic describes how to view the distribution of behavioral data and the sampling data on the dashboard.

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the R&D page that appears, move the pointer over Develop in the top navigation bar and select **Distilling**.



4. On the Distilling tab that appears, click **Behavior Engine** in the left-side navigation pane. Then, click **Dashboard** at the bottom of the left-side navigation pane.



- In the left-side pane of the **Behavior Dashboard** page, the behavioral domain, line-of-business, and object appear at three levels.
  - You can click the name of a behavioral domain or a bar under **Domain** to view the distribution of behavioral data in the behavioral domain.
  - You can click a bar under **Line-of-Business** to view the distribution of behavioral data in a line-of-business.
  - You can also click a bar under **Objects** to view the distribution of behavioral data for an action.
- In the right-side pane of the **Behavior Dashboard** page, the sampling result of the behavioral data appears. By default, the sampling result of the global behavioral data appears. If you select a behavioral domain, the sampling result of the behavioral domain appears. The sampling data is sourced from the recurring tasks corresponding to behavior rules, and the sampling data timestamp is yesterday.

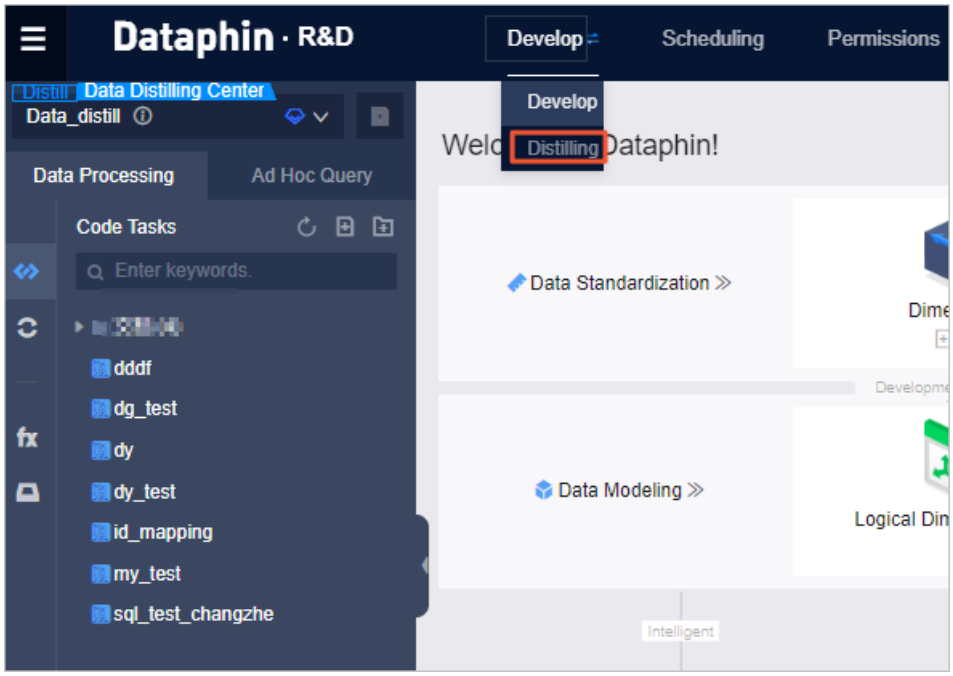
9.10.3. Tag Engine

9.10.3.1. Create a factory tag

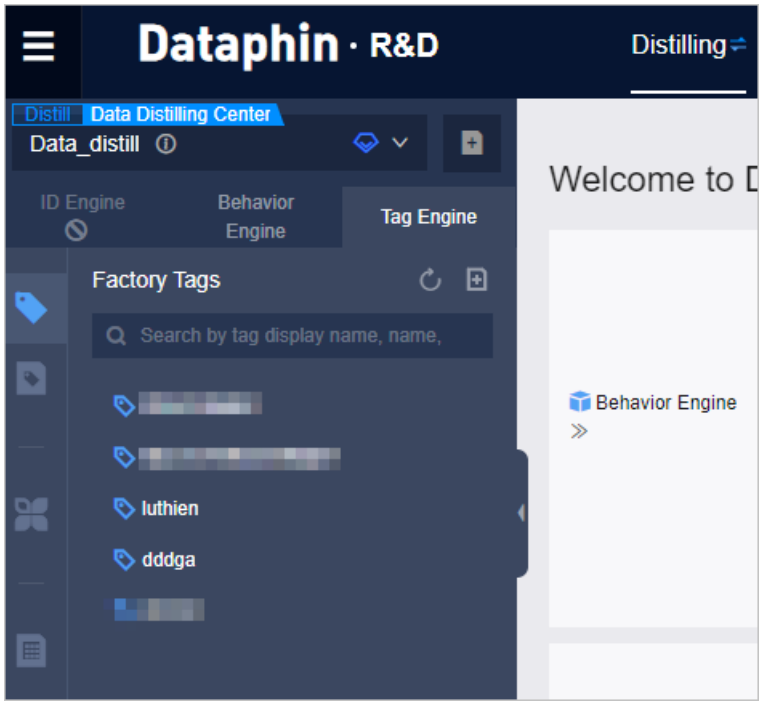
This topic describes how to create and configure a factory tag.

Create a factory tag

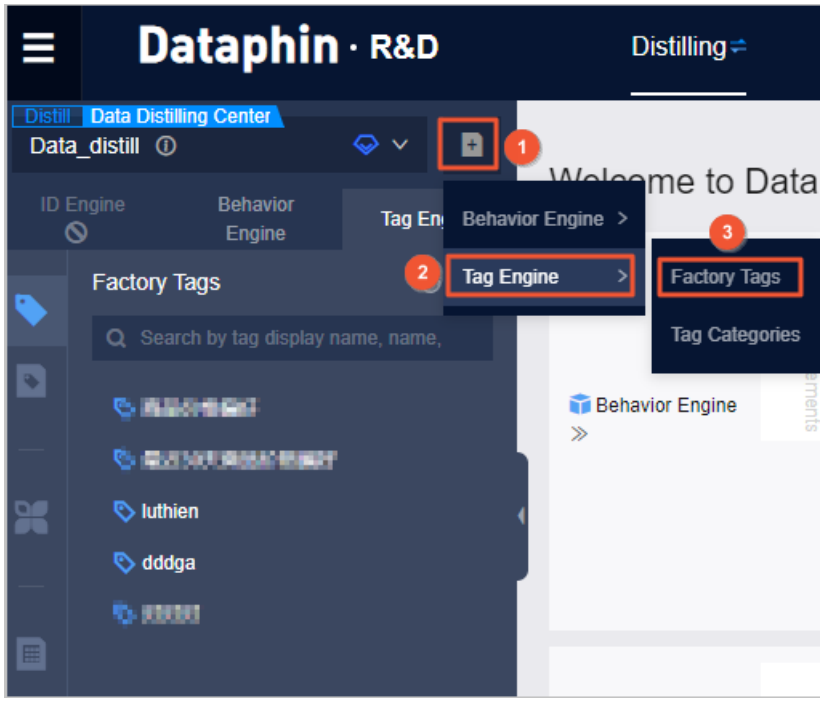
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **R&D** page that appears, move the pointer over **Develop** in the top navigation bar and select **Distilling**.



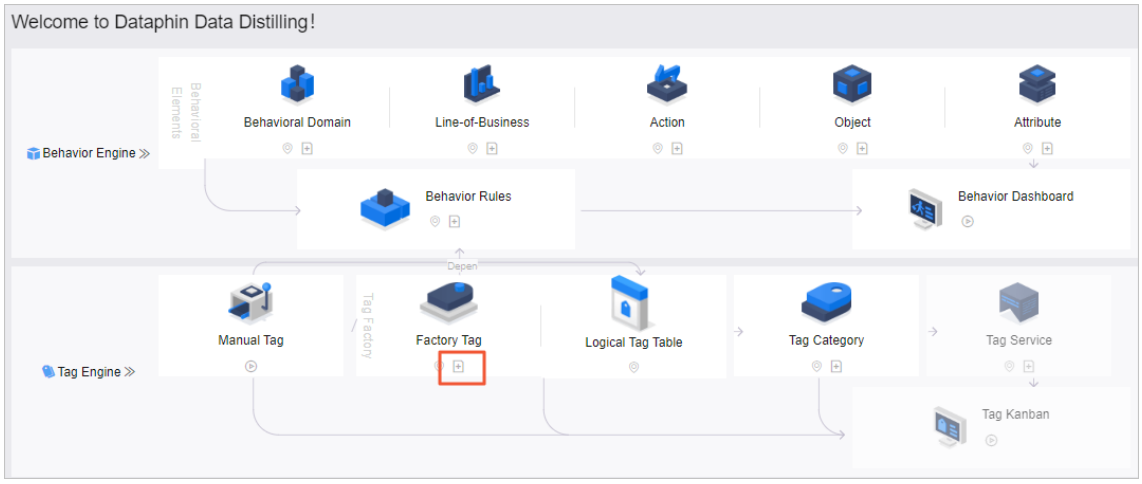
4. On the **Distilling** tab that appears, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu.



5. Open the **Create Factory Tag** dialog box in one of the following ways:
   - On the **Distilling** tab, click the icon in the upper-right corner of the left-side navigation pane.

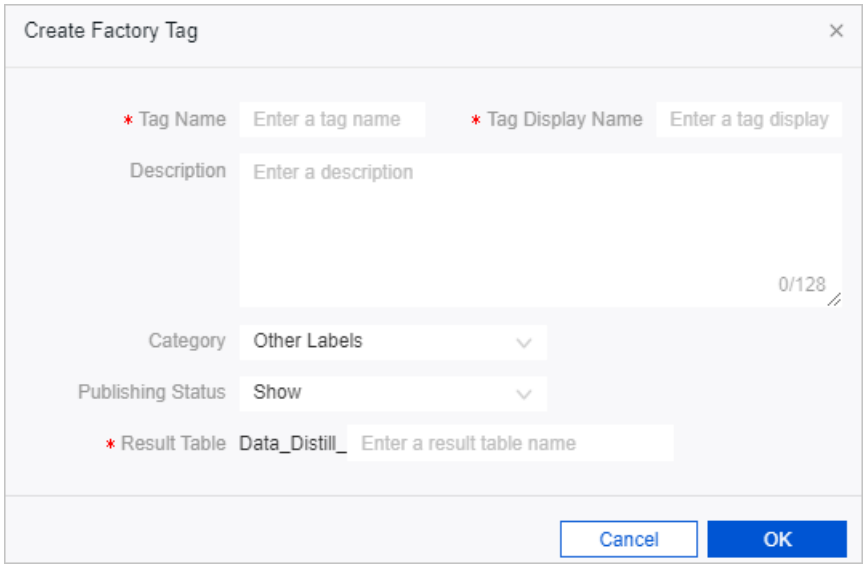


   - On the **Distilling** tab, click the icon next to the project name and choose **Tag Engine** > **Factory Tags**.
   - On the **Distilling** tab, click the icon under **Factory Tag** in the workspace.



- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, click **+ Create Tag** in the upper-right corner.

6. In the **Create Factory Tag** dialog box that appears, set the parameters as prompted and click **OK**.



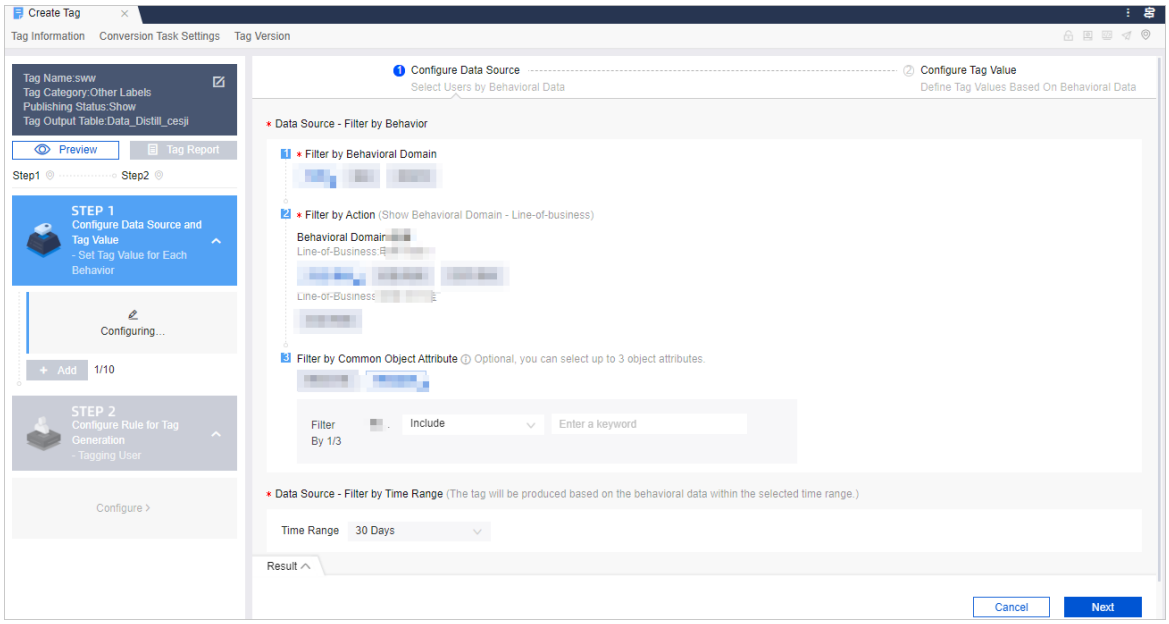
> **Note**
>
> - The value of **Tag Display Name** can contain Chinese characters, letters, digits, underscores (_), and hyphens (-). It cannot only contain digits.
> - The value of **Tag Name** can contain letters, digits, and underscores (_). It cannot only contain digits.

Configure the data source and tag values

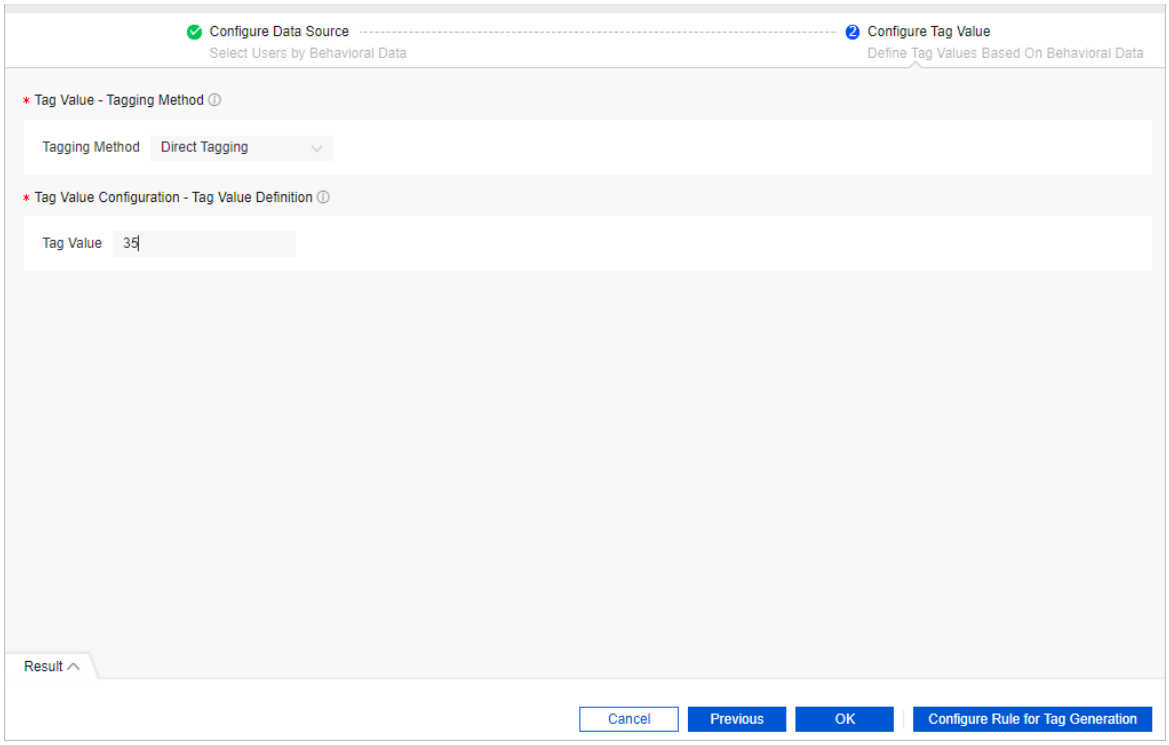
1. On the **Create Tag** tab that appears, click **Configure Data Source and Tag Value** under **Create Tag** and then click **Configure**.
2. In the **Configure Data Source** step, set the following parameters as prompted and click **Next**:
   - **Filter by Behavioral Domain**: Select the behavioral domain to which the source data

belongs. You can select multiple behavioral domains.

- **Filter by Action**: Select the actions and objects based on the line-of-business in the behavioral domain. You can select a maximum of 10 objects for all datasets. The number of actions is not limited.
- **Filter by Common Object Attribute**: If the selected behavioral data has common object attributes, you can filter the required source data based on the common object attributes. You can select a maximum of three common object attributes.
- **Data Source - Filter by Time Range**: Select the time range. Dataphin performs subsequent calculation based on the behavioral data in the selected time range to generate the final tag. The valid values are **30 Days**, **60 Days**, **90 Days**, **180 Days**, and **365 Days**.

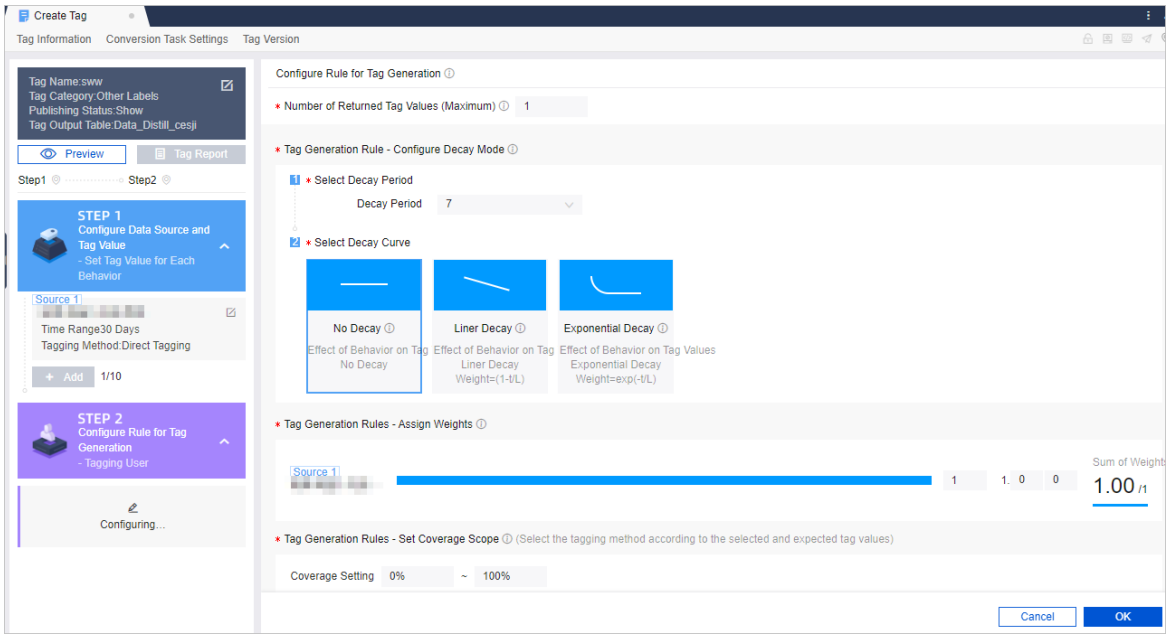


3. In the **Configure Tag Value** step, set the following parameters as prompted and click **OK**:
   - **Tag Value - Tagging Method**: the tagging method. Valid values: **Direct Tagging** and **Object Attribute Mapping**.
   - **Tag Value Configuration - Tag Value Definition**: the definition of the tag value. Set this parameter based on the value of the **Tag Value - Tagging Method** parameter.



4. Click **Configure Rule for Tag Generation** under **Configure Data Source and Tag Value** and then click **Configure**. In the **Configure Rule for Tag Generation** step, set the following parameters as prompted and click **OK**:
   - **Number of Returned Tag Values**: the maximum number of tag values that can be returned for each behavior subject ID. Each ID may generate multiple behavioral data records and therefore can have multiple tag values. Dataphin preferentially returns the tag value with the higher preference based on the number of tag values you set. You can view the tag value preference in the tag result table.
   - **Select Decay Period**: the period of the decay. The behaviors occurred in different periods have different impacts on tag values. Generally, behaviors in more recent periods have greater impacts. The behaviors occurred in the same period have the same impact on tag values. The impact of behaviors in different periods decreases with the time based on the specified decay period and curve. If the tag values change rapidly with the behavior, such as the diaper model preference, we recommend that you select a shorter period. If the tag values are stable and change slowly with the behavior, such as individual skin type prediction, we recommend that you select a longer period.
   - **Select Decay Curve**: the curve of the decay. You can set the decay mode for the behavioral data within the selected time range based on the number of segments in different decay periods. For example, if the time range is 90 days and the decay period is seven days, the number of segments is 13 (90/7 = 13). The impact of behavioral data in different periods on final tag values is related to the decay mode you select. For example, the behaviors of adding skincare products to favorites in the previous seven days and last seven days do not affect the definition of the efficacy preference for skincare products. In this case, you can select **No Decay**.
   - **Tag Generation Rules - Assign Weights**: the weights of behaviors based on the business judgment. Each weight can have a maximum of two decimal places, and the sum of weights is 1. A larger weight indicates a greater impact of this behavior on the final tag value.

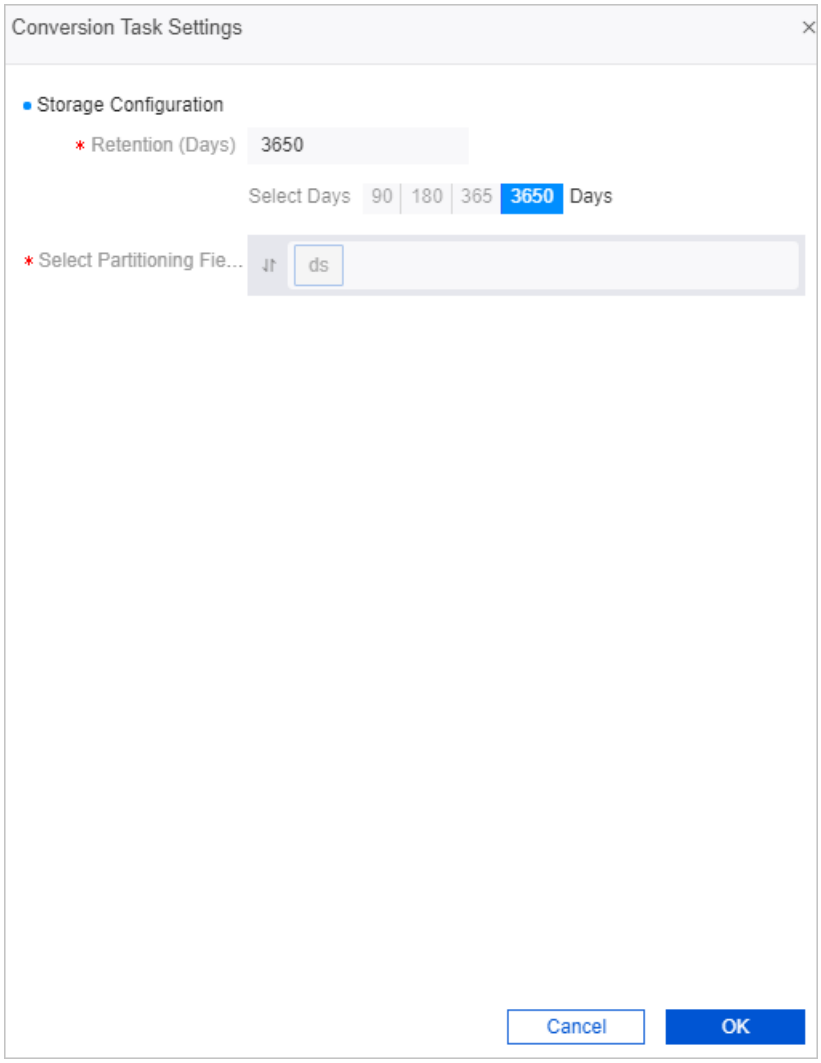
- **Tag Generation Rules - Set Coverage Scope**: the scope of the covered users.



After the preceding steps are completed, a factory tag in the **Draft** state is generated.

Configure conversion task settings

1. On the **Create Tag** tab, click **Conversion Task Settings** at the top. The **Conversion Task Settings** dialog box appears.
2. Set the parameters as prompted and click **OK**.



3. Click the icon in the upper-right corner of the page to save the configuration.
4. Click the icon in the upper-right corner of the page to test the factory tag. After the required information is specified, we recommend that you test the tag before submitting it to preview the distribution of tag values and the sampling data in the tag result table. A factory tag can be in the **Test Not Running**, **Test Running**, **Test Run Succeeded**, or **Test Run Failed** state.

   > **Note** You cannot modify the tag configuration during the test. To modify the tag configuration, stop the test first. After the test is stopped, the tag enters the **Test Run Failed** state. If you do not need to test the tag, you can directly submit the tag.

5. Click the icon in the upper-right corner of the page to submit the factory tag.

Note

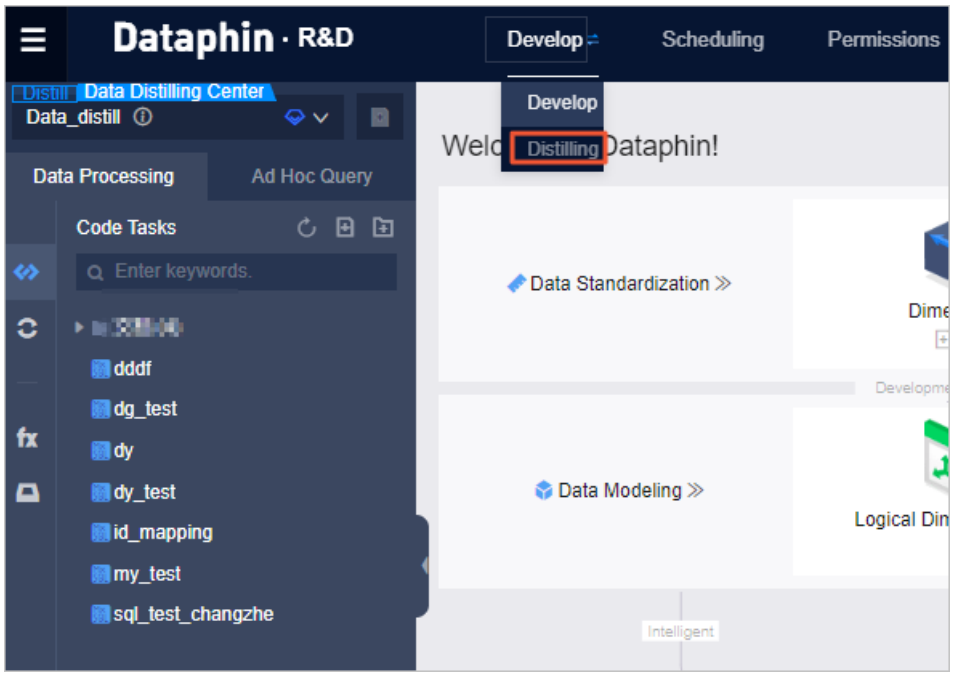
- After submitting the tag, you must generate retroactive data for the tag to obtain the historical behavioral data within the selected time range as the source data. If you do not generate retroactive data, the tag result table may be empty.
- After you submit the tag, it is published to the production environment. A recurring task is generated for scheduling.

9.10.3.2. Manage factory tags

This topic describes how to manage factory tags, for example, how to view, edit, unpublish, and unpublish and delete a factory tag, as well as how to generate retroactive data for a factory tag.

View a factory tag

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **R&D** page that appears, move the pointer over **Develop** in the top navigation bar and select **Distilling**.



4. On the **Distilling** tab that appears, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu.
5. View the information about a factory tag in different dimensions in one of the following ways:
   - In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, you can view the basic information about the tag.

On the **View Attributes** tab, click **Change** to go to the **Change Tag** tab. On the **Change Tag** tab, click **Tag Version** at the top. In the **Tag Version** dialog box that appears, you can view the versions of the tag.

On the **View Attributes** tab, click the result table name in the **Result Table** section to view the details of the result table.

On the **View Attributes** tab, click a behavior rule ID in the **Dependency** section to view the details of the behavior rule on which the tag depends.

On the **View Attributes** tab, click a behavior ID in the **Dependency** section to go to the **Distilling Object List** tab. On the **Behavior Rules** page that appears, you can view the basic information about the behavior.

- In the left-side navigation pane, click **Factory Tags Object List** at the bottom. On the **Factory Tags** page that appears in the workspace, set the filter conditions to view the information about multiple factory tags. You can also click the name of a factory tag to view the information about the tag.
  - **Development Status**: the status of the tag. The valid values are **Draft**, **Submitted**, and **Developing. Developing** indicates that the submitted tag is edited and saved but not submitted again, and the tag is involved in task scheduling in the production environment based on the previous configuration. **Submitted** indicates that the tag is involved in task scheduling in the production environment. **Draft** indicates that the new tag is not submitted.
  - **Category**: the category of the tag.
  - **Test Run Status**: the status of test run for the tag. The valid values are **Test Not Running**, **Test Running**, **Test Run Succeeded**, and **Test Run Failed**.
  - **Recent Retroactive Data**: the status of retroactive data generation for the tag. The valid values are **No Retroactive Data**, **Running**, **Succeeded**, and **Failed**. You can click the **Refresh Log** icon next to the status of retroactive data generation for a factory tag to view all the retroactive data generation records of the tag. In the **Refresh Records** window that appears, click the icon in the **Refresh Status** column of a retroactive data generation record. On the **Scheduling** tab that appears, you can view the directed acyclic graph (DAG) of the retroactive data generation instance.

Edit a factory tag

1. Go to the **Change Tag** tab in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select**Change**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, click **Change**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, click the icon in the **Actions** column of the target tag.
2. On the **Change Tag** tab that appears, set the parameters. For more information, see Create

a factory tag.

> Note You cannot modify the name of the result table if the tag is in the Submitted or Developing state.

Generate retroactive data

1. Open the **Generate Retroactive Data for Tags** dialog box in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, click the icon in the **Actions** column of the target tag to open the**Generate Retroactive Data for Tags** dialog box.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, click **Refresh**.
2. Set **Effective Period** and **Instance Name** as prompted. Then, click **OK**.

> Note You can only generate retroactive data for factory tags in the Submitted or Developing state.

View a tag task

1. Go to the **Tag Tasks** page of the **Scheduling** module in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select**Tag Tasks**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, move the pointer over the icon and select **Tag Task**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, click the icon in the **Actions** column of the target tag.

> Note You can only view the tag task for factory tags in the Submitted or Developing state.

2. On the **Tag Tasks** page of the **Scheduling** module that appears, view and maintain the production data.

View a tag report

1. View the report of a factory tag in one of the following ways:

- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select**Tag Report**.
- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, click **View Tag Report**.
- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane.

  On the **Factory Tags** page that appears in the workspace, if the development status of the target tag is **Submitted** or **Developing**, move the pointer over the icon in the **Actions** column and select **Tag Report**.

  If the development status of the target tag is **Draft**, click the icon in the **Actions** column.

2. On the tag report details tab that appears, view the information about the tag report.

Clone a factory tag

1. Go to the **Clone Tag** tab in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select**Clone**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, move the pointer over the icon and select **Clone**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, if the development status of the target tag is **Submitted** or **Developing**, move the pointer over the icon in the **Actions** column and select**Clone**.

     If the development status of the target tag is **Draft**, click the icon in the **Actions** column.

2. On the **Clone Tag** tab that appears, view the detailed configuration information about the cloned tag. You can also modify the configuration based on your business requirements. For more information about how to modify the configuration, see Edit a factory tag.

Unpublish a factory tag

1. Open the **Tip** dialog box to unpublish a factory tag in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select**Unpublish**.

- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, move the pointer over the icon and select **Unpublish**.
- On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, move the pointer over the icon in the **Actions** column of the target tag and select**Unpublish**.

2. In the **Tip** dialog box that appears, click **OK**. After a factory tag is unpublished, it enters the **Draft** state.

> **Note** You can only unpublish factory tags in the **Submitted** or **Developing** state.

Unpublish and delete a factory tag

1. Open the **Tip** dialog box to unpublish and delete a factory tag in one of the following ways:
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side factory tag list, move the pointer over the icon next to the target tag and select **Unpublish and Delete**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. In the left-side factory tag list, click the target tag. On the **View Attributes** tab that appears, move the pointer over the icon and select **Unpublish and Delete**.
   - On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click **Factory Tags** on the left-side navigation submenu. Then, click **Factory Tags Object List** at the bottom of the left-side navigation pane. On the **Factory Tags** page that appears in the workspace, move the pointer over the icon in the **Actions** column of the target tag and select **Unpublish and Delete**.
2. In the **Tip** dialog box that appears, enter the comments and click **OK**.

9.10.3.3. Manage the logical tag table

This topic describes how to manage the logical tag table, for example, how to view the logical tag table and modify the factory tags and manual tags in the logical tag table.

Dataphin automatically creates a logical tag table when a distilling project is initialized. The logical tag table collects all the factory tags that are in the Submitted state and manual tags in physical tables by id_type and id_value. Note that manual tags in physical tables are not developed in the tag factory. The name of the logical tag table is fixed to Ads_distill_labels and cannot be changed.

View the logical tag table

1. Log on to the Dataphin console.
2. Go to the **Logical Tag Tables** section.

i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the Develop page, move the pointer over **Develop** in the top navigation bar and select **Distilling**.

iii. On the **Distilling** page, click the **Tag Engine** tab in the left-side navigation pane.

iv. On the **Tag Engine** tab, click the icon on the left-side navigation submenu. The **Logical Tag Tables** section appears.

3. In the **Logical Tag Tables** section, click **Ads_distill_labels**. The **Logical Tag Table** tab appears.

   The **Logical Tag Table** tab consists of the **Factory Tags** and **Manual Tags** tabs. All the factory tags in the **Submitted** or **Developing** state are automatically attached to the logical tag table. You can also import manual tags that are developed by coding from physical tables to the logical tag table.

Modify a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.
2. On the **Factory Tags** tab, find the factory tag that you want to modify by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Click the icon in the **Actions** column.
4. On the **Change Tag** tab, modify the parameters as required. For more information, see Edit a factory tag.

Generate retroactive data for a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.
2. On the **Factory Tags** tab, find the factory tag for which you want to generate retroactive data by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Click the icon in the **Actions** column.
4. In the **Generate Retroactive Data for Tags** dialog box, set the parameters as required. For more information, see Generate retroactive data.

View the tag task under a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.

2. On the **Factory Tags** tab, find the factory tag whose tag task you want to view by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Click the icon in the **Actions** column.
4. In the **Tag Tasks** section on the Distilling Management tab, view the details about the tag task under the factory tag.

Clone a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.
2. On the **Factory Tags** tab, find the factory tag that you want to clone by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Move the pointer over the icon in the **Actions** column and select **Clone**.
4. On the **Clone Tag** tab, modify the parameters as required. For more information, see Edit a factory tag.

Unpublish a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.
2. On the **Factory Tags** tab, find the factory tag that you want to unpublish by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Move the pointer over the icon in the **Actions** column and select **Unpublish**.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Unpublish and delete a factory tag

1. On the **Logical Tag Table** tab, click **Factory Tags**.

2. On the **Factory Tags** tab, find the factory tag that you want to unpublish and delete by using one of the following methods:
   - Click the icon. Then, filter factory tags by setting the **Category**, **Development Status**, **Test Run Status**, and **Last Refresh Status** parameters.
   - Enter a tag display name, tag name, or tag ID in the search box to search for your desired factory tag.
   - Find your desired factory tag in the tag list.
3. Move the pointer over the icon in the **Actions** column and select **Unpublish and Delete**.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

Create a manual tag

1. On the **Logical Tag Table** tab, click **Manual Tags**.
2. On the **Manual Tags** tab, click **Create Manual Tags** in the upper-right corner.
3. In the **Create Manual Tag** dialog box, perform the following steps:
   i. In the **Association Logic** step, set the **Source Physical Table** parameter and parameters in the **Association Logic** section.
   ii. Click **OK**.
   iii. In the **Add Fields** step, select the fields that are used to associate the source physical table with the logical tag table and click **Add >>**.
   iv. On the **New List** tab, set the **Tag Display Name** parameter.

   You can also modify the **Tag Name**, **Category**, and **Publishing Status** parameters. To delete a field from the New List tab, click the icon in the Actions column. In the message that appears, click **OK**.
   v. Click **OK**.
4. In the **Description** dialog box, enter your comments and click **OK**.

Modify a manual tag

1. On the **Logical Tag Table** tab, click **Manual Tags**.
2. On the **Manual Tags** tab, find the manual tag that you want to modify by using one of the following methods:
   - Filter manual tags by setting the **Category** parameter.
   - Enter a keyword in the search box to search for manual tags whose names contain the keyword.
   - Find your desired manual tag in the tag list.
3. Click the icon in the **Actions** column.
4. In the **Modify Manual Tag** dialog box, set the parameters as required.

Parameter	Description
Tag Name	The name of the manual tag. You cannot change the name.
Tag Display Name	The display name of the manual tag. You cannot change the display name.
Description	The description of the manual tag.
Category	The category of the manual tag.
Publishing Status	Specifies whether to show the manual tag.

5. Click **Submit**.
6. In the **Description** dialog box, enter your comments.
7. Click **OK**.

Delete a manual tag

1. On the **Logical Tag Table** tab, click **Manual Tags**.
2. On the **Manual Tags** tab, find the manual tag that you want to delete by using one of the following methods:
   - Filter manual tags by setting the **Category** parameter.
   - Enter a keyword in the search box to search for manual tags whose names contain the keyword.
   - Find your desired manual tag in the tag list.
3. Click the icon in the **Actions** column.
4. In the **Tip** dialog box, enter your comments.
5. Click **OK**.

9.10.3.4. Manage tag categories

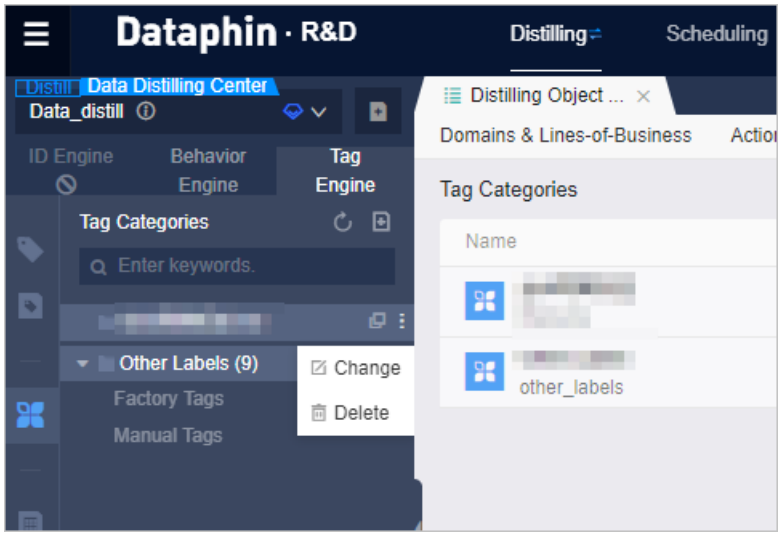
This topic describes how to create, edit, and delete a tag category.

Create a tag category

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar.
3. On the **R&D** page that appears, move the pointer over **Develop** in the top navigation bar and select **Distilling**. On the **Distilling** tab, click **Tag Engine** in the left-side navigation pane and click the icon on the left-side navigation submenu. In the left-side navigation pane, click the icon. In the **Create Category** dialog box that appears, set the parameters as prompted and click **OK**.

Edit a tag category

1. In the left-side tag category list, move the pointer over the icon next to the target category and select **Change**.



2. In the **Modify Category** dialog box that appears, set the parameters as prompted and click **OK**.

Delete a tag category

In the left-side tag category list, move the pointer over the icon next to the target category and select **Delete**. In the **Tip** dialog box that appears, enter the comments and click **OK**. Then, the tag category is deleted.

9.11. Task publishing

9.11.1. Publishing management

In Dev-Prod mode, to schedule data modeling, data processing, and data integration tasks that are generated in the development environment, you must publish them to the production environment. This topic describes how to publish tasks and view the publishing records.

- In Dev-Prod mode, tasks rest on the Objects to Publish page after you submit them. You must publish them before you can schedule and manage them in the production development.
- In Basic mode, tasks are directly submitted to the production environment.

> **Note** The publishing feature is available only in the development environment.

Objects to Publish page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar. The Develop page appears.
3. Click **Publish** in the top navigation bar. The **Objects to Publish** page appears.

The **Objects to Publish** page displays tasks and files on four tabs: **Pipeline Script**, **Standards and Modeling**, **Batch Processing**, and **Stream Processing**. Tasks include stream processing tasks, batch processing tasks, and sync tasks. Files include resources, batch processing functions, stream processing functions, stream metatables, and stream processing templates.

On the **Objects to Publish** page, you can click the **Pipeline Script**, **Standards and Modeling**, **Batch Processing**, or **Stream Processing** tab to view objects to be published as required. The four tabs provide similar features. This section uses the **Standards and Modeling** tab as an example.You can filter objects by last submitter, last submission time, object type, and operation. You can also enter a keyword in the search box to search for objects whose names contain the keyword.

Parameter	Description
Object Name	The name of the object.
Object ID	The ID of the object.
Object Type	The type of the object.
Version	The version of the object.
Operation	The operation that was performed on the object. Valid values: ○ **Add** ○ **Update** ○ **Delete**
Last Submitted By	The user who submitted the object last time and the submission time.
Actions	○ a. Click the icon in the **Actions** column to publish the object. You can also select multiple objects and click **Publish** in the lower part of the page to publish them at a time. b. In the **Publish** dialog box, click **OK** to publish one or more objects. ○ Click the icon in the **Actions** column to view the publishing records of the object. ○ Click the icon in the **Actions** column to edit the object on the page that appears.

Publishing History page

In the left-side navigation pane, click **Publishing History**. On the **Publishing History** page, you can search for published objects and view the publishing status and details of a specific object. You can also click the Edit icon in the Actions column of an object to edit the object on the page that appears.You can filter objects by last publisher, last publishing time, object type, operation, and publishing status. You can also enter a keyword in the search box to search for objects whose names contain the keyword.

Parameter	Description
Publish Name	The publishing name of the object.
Publish ID	The publishing ID of the object.
Object Name	The name of the object.
Object ID	The ID of the object.
Object Type	The type of the object.
Node ID	The task ID of the object.
Version	The version of the object.
Operation	The operation that was performed on the object. Valid values: • **Add** • **Update** • **Delete**
Published By/Start Publishing At	The user who published the object and the publishing time.
Publishing Status/Finish Publishing At	The publishing status of the object and the time when the publishing was completed. Publishing states include: • **Publish Failed** • **Published Successfully** • **Publishing**
Actions	• Click **Details**. In the **Publishing Details** dialog box, view the publishing details. • Click the icon in the **Actions** column to edit the object on the page that appears. • Click the icon in the **Actions** column. On the **Objects to Publish** page, view the objects that failed to be published. Note that objects that failed to be published will rest on the **Objects to Publish** page. You can publish them again.

9.12. Scheduling center

9.12.1. Overview

Dataphin supports policy-based scheduling and management of code tasks that are generated by data modeling, coding, and data distilling. The scheduling and management feature includes global management, logical table management, and distilling management.

Scheduling page

The Scheduling page contains the Global Management tab. You can go to the Global Management tab by performing the following steps:

1. Log on to the Dataphin console.
2. Go to the **Global Management** tab by using one of the following methods:
   - If the project that you accessed last time is in **Basic** or **Prod** mode, use one of the following methods to go to the **Global Management** tab:
     - a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner.
       b. On the Scheduling page, click the icon next to the project name in the upper-left corner, click the **Dev** tab, and then select a project. The **Global Management** tab appears by default.
     - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
       b. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev** tab, and then select a project.
       c. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.
   - If the project that you accessed last time is in **Dev** mode, use one of the following methods to go to the **Global Management** tab:
     - On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner. The **Global Management** tab appears by default.
     - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
       b. On the **Develop** page, click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

In Dev-Prod mode, the scheduling tasks in projects in Dev mode are independent of those in the corresponding projects in Prod mode.

- In projects in Dev mode, the Scheduling page contains the **Global Management** and **Logical Table Management** tabs.
- The scheduling tasks in projects in Dev mode cannot automatically run. You must manually run the scheduling tasks to generate instances.
- In projects in Prod mode, the Scheduling page contains the **Global Management**, **Logical Table Management**, and **Monitoring** tabs. You can configure alert rules to monitor tasks only in projects in Prod mode.
- Dataphin generates and runs scheduling tasks in projects in Prod mode based on scheduling policies.
- You can click the Prod Dev icon next to the project name in the upper-left corner to switch between the Scheduling page of a project in Dev mode and that of the corresponding project in Prod mode.

  For example, if you click the Prod Dev icon on the Scheduling page of the test_dev project, the Scheduling page of the test project appears.

In projects in Basic mode:

- The Scheduling page contains the **Global Management**, **Logical Table Management**, and **Monitoring** tabs.
- Dataphin generates and runs scheduling tasks in projects in Basic mode based on scheduling policies. You can configure alert rules to monitor tasks in projects in Basic mode.

Tasks

A task is an object that can be scheduled in the production environment after you submit and publish its code and configuration in the development environment. A task can be scheduled at specified intervals or manually triggered. Tasks are classified into the following types based on the way they are run:

- **Recurring tasks**: the tasks for which a recurrence is configured. The scheduling system of Dataphin schedules recurring tasks at specified intervals and generates corresponding recurring instances.
- **One-time tasks**: the tasks that are not automatically triggered by the scheduling system. You can manually run a one-time task as needed. Each time you run a one-time task, a one-time instance is generated.
- **Stream processing tasks**: the tasks for processing streaming data. You must manually run production or test instances for stream processing tasks.

The scheduling sequence of a task is determined based on the recurrence, dependency, and priority that are configured for the task.

- To apply the code or configuration update of a task to the instance generated the next day, you must update the code or configuration before 23:00 on the current day.
- When an instance runs, Dataphin reads the latest code and scheduling configuration of the corresponding task.
- You can specify a data timestamp within the range from 1970 to 2099 for Dataphin to generate retroactive data. Each scheduling instance is retained for up to two weeks.

Instances

An instance is generated each time a task is run. Instances can be generated in one of the following ways:

- Recurring instances are automatically and periodically generated when recurring tasks are scheduled by the scheduling system.
- Retroactive data generation instances are generated when you generate retroactive data for tasks.
- One-time instances are generated when you run one-time tasks.
- Stream processing instances are generated when you run production or test instances for stream processing tasks.

Logical tables

Logical table management involves logical table tasks and logical table task instances.

- The **Logical Table Tasks** section displays the tasks of logical tables.
- The **Logical Table Task Instances** section displays the instances of logical table tasks that are run and the states of these instances.

Distilling management

Distilling management involves tasks and instances. Distilling management allows you to manage tasks and instances related to behavior rules and tags from a business standpoint.

9.12.2. Global management

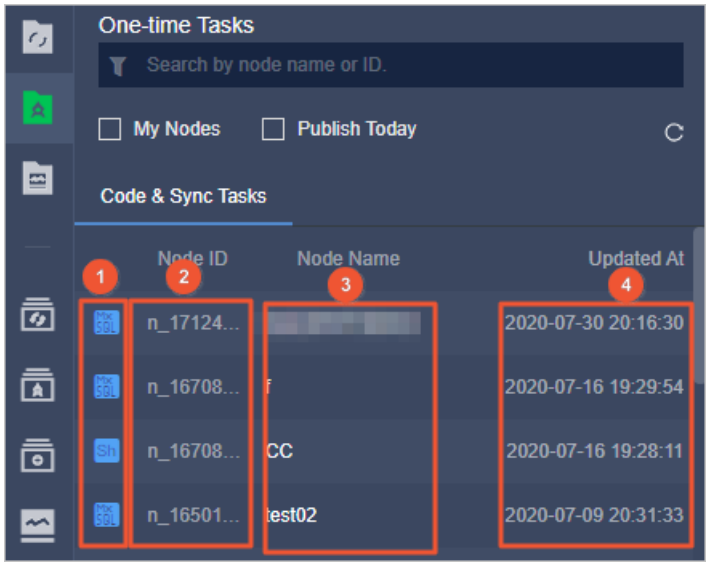
9.12.2.1. One-time tasks

The One-time Tasks section on the Global Management tab displays the one-time tasks that have been submitted and published. This topic describes the operations that you can perform on one-time tasks. For example, you can view, run, and modify one-time tasks.

Go to the One-time Tasks section

To go to the One-time Tasks section, perform the following steps:

1. Log on to the Dataphin console.
2. Go to the **Global Management** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.
   iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.
3. Click the icon on the left-side navigation submenu. The **One-time Tasks** section appears.



No.	Description
1	The type of the task.
2	The node ID of the task.
3	The node name of the task.

No.	Description
4	The time when the task was last modified.

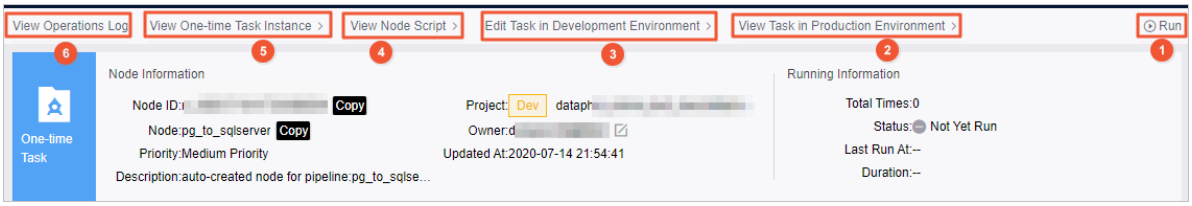
Search for one-time tasks

- Click the icon in the search box, click the icon next to **Owner** or **Task Type**, and then select an owner or a task type. You can also select both an owner and a task type to search for your desired one-time tasks.
- Enter a keyword in the search box to search for one-time tasks whose names contain the keyword.

Manage a one-time task in the right-side workspace

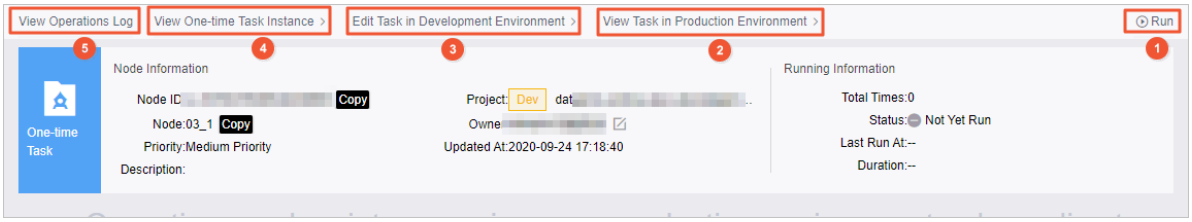
Click a one-time task in the **One-time Tasks** section. The task details appear in the right-side workspace.

- The following figure shows the workspace of a one-time task that is submitted or published from the **Integrated** module.



No.	Description
1	To run a one-time task, perform the following steps: i. Click **Run** in the upper-right corner. ii. In the **Run** dialog box, modify the **Instance Name** and **Data Timestamp** parameters or use their default values. iii. Click **OK**.
2	Click **View Task in Production Environment** to view the code of the task in the production environment.
3	Click **Edit Task in Development Environment** to edit the task in the development environment.
4	Click **View Node Script** to view the code of the task in the development environment.
5	Click **View One-time Task Instance**. In the One-time Task Instances section, view the instances that have been generated for the task.
6	Click **View Operations Log**. In the **Operations Log** pane, view the operation logs of the task.

- The following figure shows the workspace of a one-time task that is submitted or published from the **Develop** module.



No.	Description
1	To run a one-time task, perform the following steps: i. Click **Run** in the upper-right corner. ii. In the **Run** dialog box, modify the **Instance Name** and **Data Timestamp** parameters or use their default values. iii. Click **OK**.
2	Click **View Task in Production Environment** to view the code of the task in the production environment.
3	Click **Edit Task in Development Environment** to edit the task in the development environment.
4	Click **View One-time Task Instance**. In the One-time Task Instances section, view the instances that have been generated for the task.
5	Click **View Operations Log**. In the **Operations Log** pane, view the operation logs of the task.

9.12.2.2. Recurring tasks

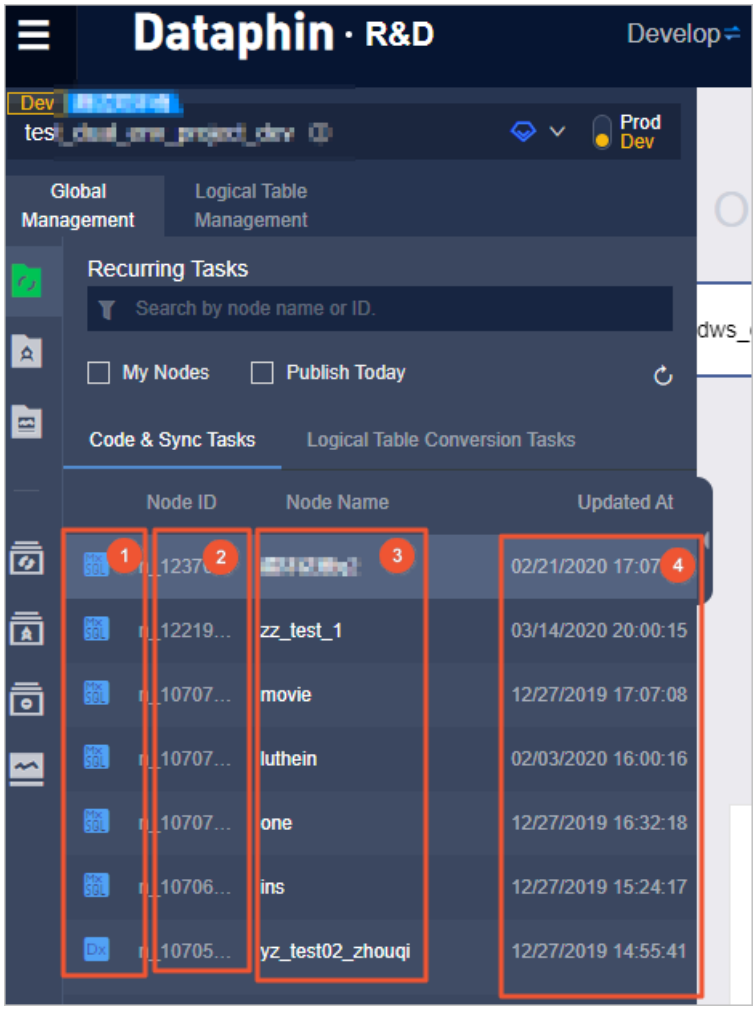
The Recurring Tasks section displays the recurring tasks that have been submitted and published. This topic describes the operations that you can perform on recurring tasks. For example, you can view recurring tasks and use DAGs to manage these tasks.

Go to the Recurring Tasks section

To go to the Recurring Tasks section, perform the following steps:

1. Log on to the Dataphin console.
2. Go to the **Global Management** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.
   iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

Click the icon on the left-side navigation submenu. The **Recurring Tasks** section appears.



No.	Description
1	The type of the task.
2	The node ID of the task.
3	The node name of the task.
4	The time when the task was last modified.

Search for recurring tasks

- Click the icon in the search box, click the icon next to **Owner** or **Task Type**, and then select an owner or a task type. You can also select both an owner and a task type to search for your desired recurring tasks.
- Enter a keyword in the search box to search for recurring tasks whose names contain the keyword.

Manage a recurring task in the right-side workspace

Click a recurring task in the **Recurring Tasks** section. The DAG of the task appears in the right-side workspace. Right-click the task node in the DAG. On the shortcut menu that appears, you can select menu items to perform supported operations.

- The following figure shows the shortcut menu for a recurring task that is submitted or published from the **Integrated** module.

Menu item	Description
Show Parent Nodes	Select **Show Parent Nodes** and select the layer that you want to view.
Show Child Nodes	Select **Show Child Nodes** and select the layer that you want to view.
View Node Script	Select **View Node Script**. On the Node Script page, view the code of the task.
View Script	Select **View Script**. On the **Pipeline Details** page, view the pipeline script of the task.
View Operations Log	Select **View Operations Log**. In the **Operations Log** pane, view the operation details of the task.
Change Owner	Select **Change Owner**. In the Change Owner dialog box, select a member to whom you want to transfer the ownership of the task. Note You can transfer a task only to a business unit administrator, the super administrator, or a project administrator.
View Instances	Choose **View Instances** > **View Recurring Instances**. In the **Recurring Task Instances** section, view the recurring instances that have been generated for the task.
	Choose **View Instances** > **View Retroactive Data Generation Instances**. In the **Retroactive Data Generation Instances** section, view the retroactive data generation instances that have been generated for the task.
Generate Retroactive Data	To generate retroactive data for the task, perform the following steps: i. Select **Generate Retroactive Data**. ii. In the **Generate Retroactive Data** dialog box, set the **Instance Name**, **Data Timestamp**, and Select Downstream Nodes parameters. iii. Click **OK**.

- The following figure shows the shortcut menu for a recurring task that is submitted or published from the **Develop** module.

Menu item	Description
Edit Node in Development Environment	Select **Edit Node in Development Environment** to edit the task in the development environment.

Menu item	Description
View Node in Production Environment	Select **View Node in Production Environment** to view the task in the production environment.
Show Parent Nodes	For more information, see the preceding table.
Show Child Nodes	
View Node Script	
View Operations Log	
Change Owner	
View Instances	Choose **View Instances** > **View Recurring Instances**. In the **Recurring Task Instances** section, view the recurring instances that have been generated for the task.
	Choose **View Instances** > **View Retroactive Data Generation Instances**. In the **Retroactive Data Generation Instances** section, view the retroactive data generation instances that have been generated for the task.
Generate Retroactive Data	To generate retroactive data for the task, perform the following steps: i. Select **Generate Retroactive Data**. ii. In the **Generate Retroactive Data** dialog box, set the **Instance Name**, **Data Timestamp**, and Select Downstream Nodes parameters. iii. Click **OK**.

9.12.2.3. Stream processing tasks

The Stream Processing Tasks section displays the stream processing tasks that have been submitted and published. These tasks must be run manually. This topic describes the operations that you can perform on stream processing tasks.

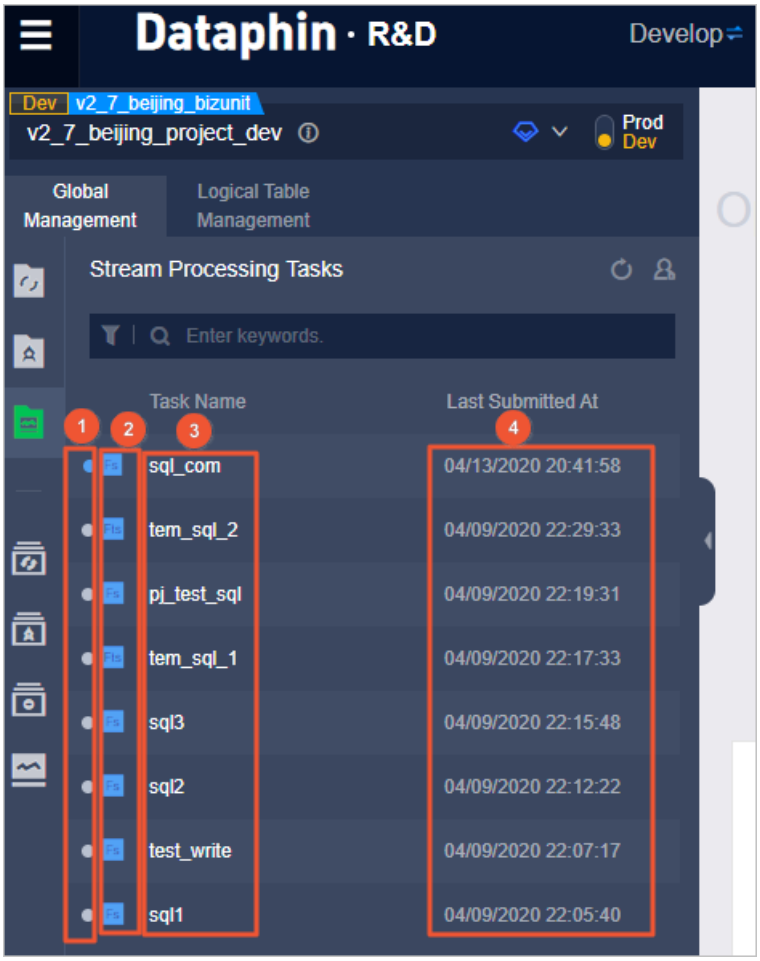
Go to the Stream Processing Tasks section

To go to the Stream Processing Tasks section, perform the following steps:

1. Log on to the Dataphin console.
2. Go to the **Global Management** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.

iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

Click the icon on the left-side navigation submenu. The **Stream Processing Tasks** section appears.



No.	Description
1	The status of the task. When you run a stream processing task in a project in Prod or Basic mode, a production instance is run. • : A production instance is running. • : No production instance is running. When you run a stream processing task in a project in Dev mode, a test instance is run. • : A test instance is running. • : No test instance is running.

No.	Description
2	The type of the task. Each task type is indicated by a unique icon. • : FLINK_SQL • : FLINK_TEMPLATE_SQL
3	The name of the task. The task name is the name of the stream processing task that you submitted.
4	The time when the task was last modified. By default, stream processing tasks are sorted in descending order of running time.

Search for stream processing tasks

- In the Stream Processing Tasks section, click the icon and then the icon. In the drop-down list that appears, select **FLINK_SQL** or **FLINK_TEMPLATE_SQL** to filter tasks of the specified type.
- Enter a task name in the search box to search for the specific task. Alternatively, enter a keyword in the search box to search for tasks whose names contain the keyword.

View stream processing tasks

Click the icon next to Stream Processing Tasks to view the stream processing tasks that you own.

Manage a stream processing task in the right-side workspace

Click a stream processing task in the **Stream Processing Tasks** section. The task details appear in the right-side workspace.

If a stream processing task in a project in Dev mode has no running test instance, click **Start Test Instance** in the upper-right corner of the workspace.

1. In the **Test Instance Parameter Configuration** dialog box, set the parameters as required.
2. ○ If you select **Print Log** for **Test Method**, set **Start Time for Reading Data**. Then, click **Start** to run a test instance.
   ○ If you select **Generate Table** for **Test Method**, set **Start Time for Reading Data** and specify the output table. Then, click **Start** to run a test instance.

The DAG displays information about the nodes of a stream processing task and relationships between the nodes. After you click a node in the DAG, the node details appear in the lower-left corner of the workspace.

You can perform the following operations on the task in the workspace:

- Click **Task Script** in the top navigation bar of the workspace. In the Task Scripts pane, view the code of the task. You can also click **Go to R&D Workbench and Edit Task Code** to edit the code.
- Click **Task Parameters** in the top navigation bar of the workspace. In the Task Parameters pane, view the parameters of the task. You can also click **Go to R&D Workbench and Edit Task Parameters** to modify the parameters.
- Click **Operations Log** in the top navigation bar of the workspace. In the Operations Log pane,

view the operation logs of the task.

- If the task belongs to a project in Dev mode, click **View Test Instance** in the top navigation bar of the workspace to view the test instance in the **Stream Processing Instances** section.
- If the task belongs to a project in Prod or Basic mode, click **View Production Instances** in the top navigation bar of the workspace to view the production instance in the **Stream Processing Instances** section.

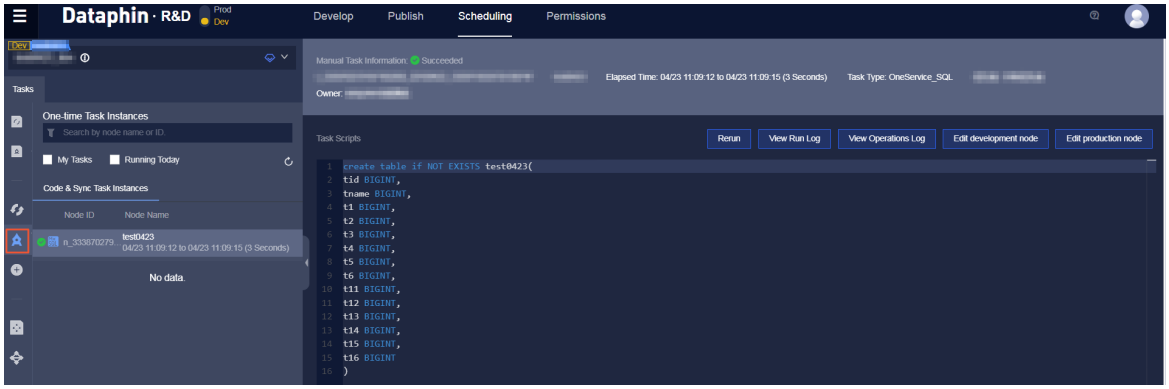
- If the task type is **FLINK_TEMPLATE_SQL**, click **Template Details** in the top navigation bar of the workspace. In the Template Details pane, view the template information. You can also click **Go to R&D Workbench to Edit Template** to edit the template information.

9.12.2.4. One-time instances

This topic describes the operations that you can perform on one-time instances.

One-time Task Instances page

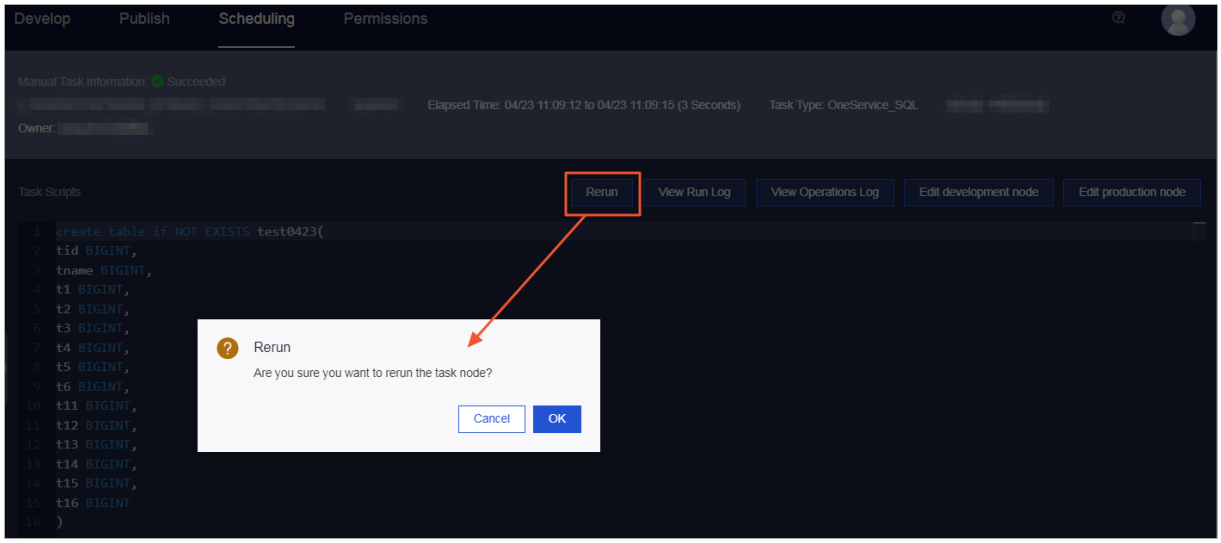
1. Go to the scheduling center. In the left-side navigation pane, click Global Management. On the left-side navigation submenu, click **One-time Task Instances**. The One-time Task Instances page appears.



2. In the left-side navigation pane of the One-time Task Instances page, view the one-time instance nodes that are sorted in descending order by running time.
   - In the left-side navigation pane, you can view the instance nodes and search for a specific instance node. In the search box at the top of the left-side navigation pane, you can enter a node name or ID to search for an instance node. You can also select the **My Task Instances** and **Running Today** check boxes to filter the instance nodes that you own and the instance nodes created on the current day, respectively. In the Code & Sync Task Instances section, all instance nodes that meet the search conditions are sorted in descending order by running time. By default, the task type (displayed as an icon), ID, name, start time, end time, and duration of an instance node are displayed in each row.
   - In the right-side workspace, you can view the code and details of the current one-time instance node, as well as the operations that can be performed on the current one-time instance node. The instance details are available at the top of the workspace, including the status, ID, name, running time, task type, and owner of the current instance node. You can rerun the instance, terminate the instance, view the operational logs, view the operations logs, and edit the node.

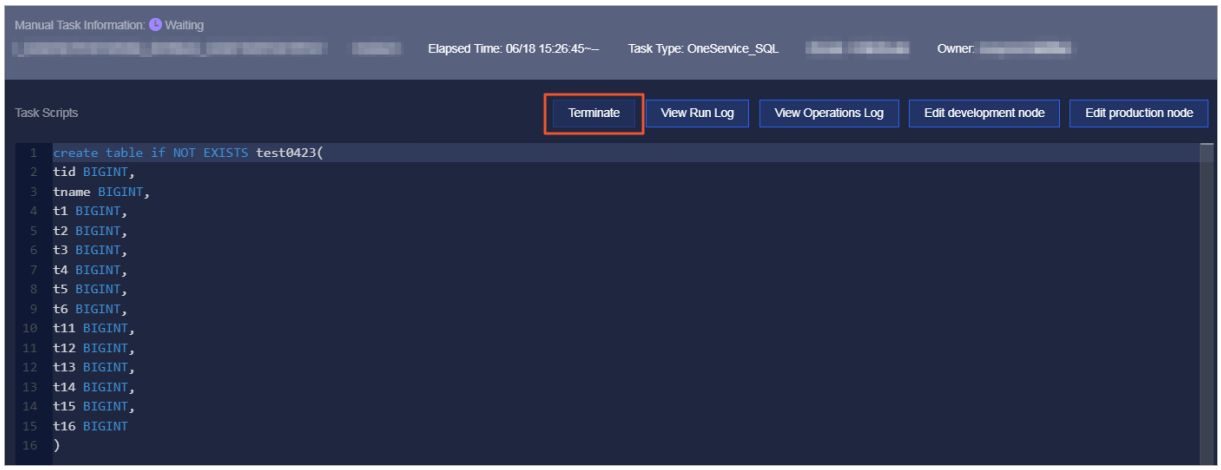
Rerun a one-time instance

As shown in the following figure, click **Rerun** in the upper-right corner of the workspace. In the dialog box that appears, click **OK**. Then, the current one-time instance is rerun.

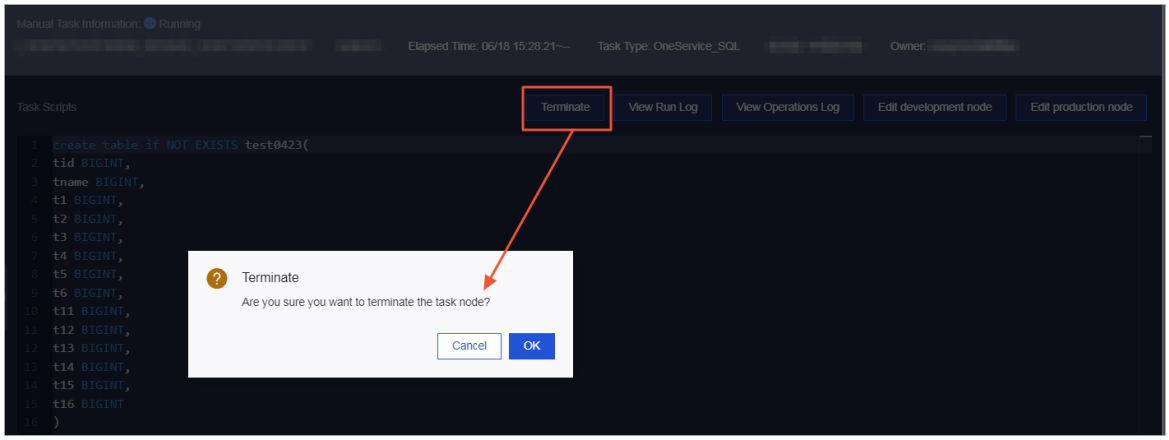


Note You can rerun a one-time instance only when the instance is completed, that is, the instance is in the Succeeded, Failed, or Not Yet Run state. Otherwise, the **Rerun** button is invisible.

Terminate a running one-time instance

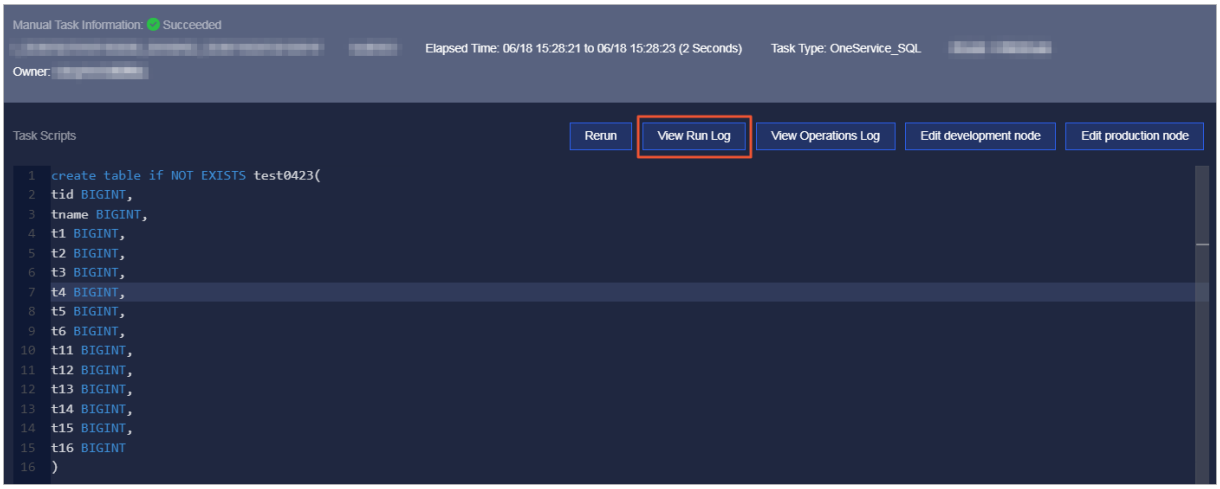


1. As shown in the preceding figure, if the current one-time instance is not completed, that is, the instance is in the Initial, Waiting, or Running state, the **Terminate** button is visible in the upper-right corner of the workspace.
2. Click **Terminate**. The Terminate dialog box appears, as shown in the following figure.

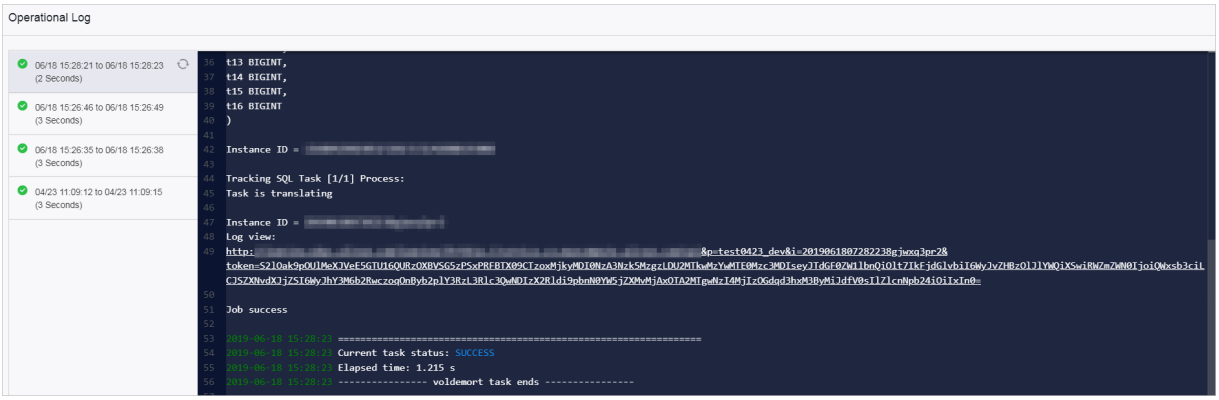


3. Click **OK**. Then, the current running instance is terminated. After a one-time instance is terminated, its final status is Failed.

View the operational logs of an instance

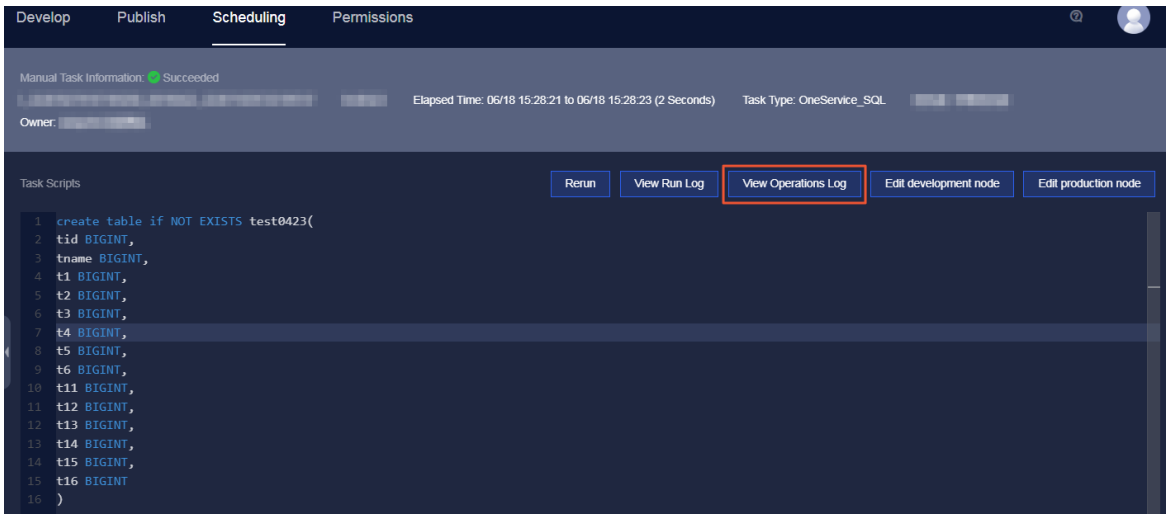


As shown in the preceding figure, click **View Operational Logs** at the top of the workspace. On the Operational Log page that appears, you can view the detailed running process of the current one-time instance, as shown in the following figure.

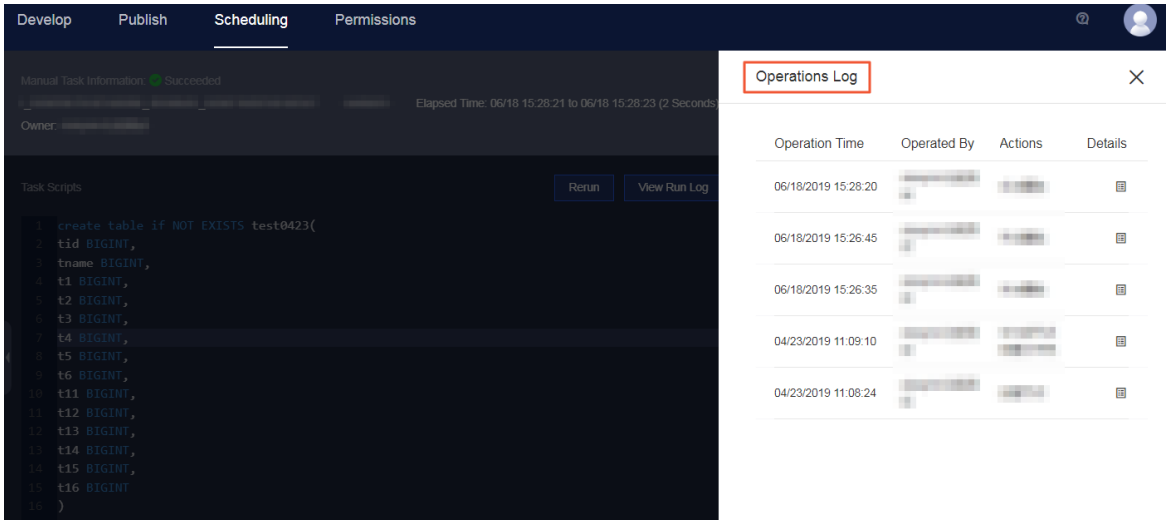


View the operations logs of a task

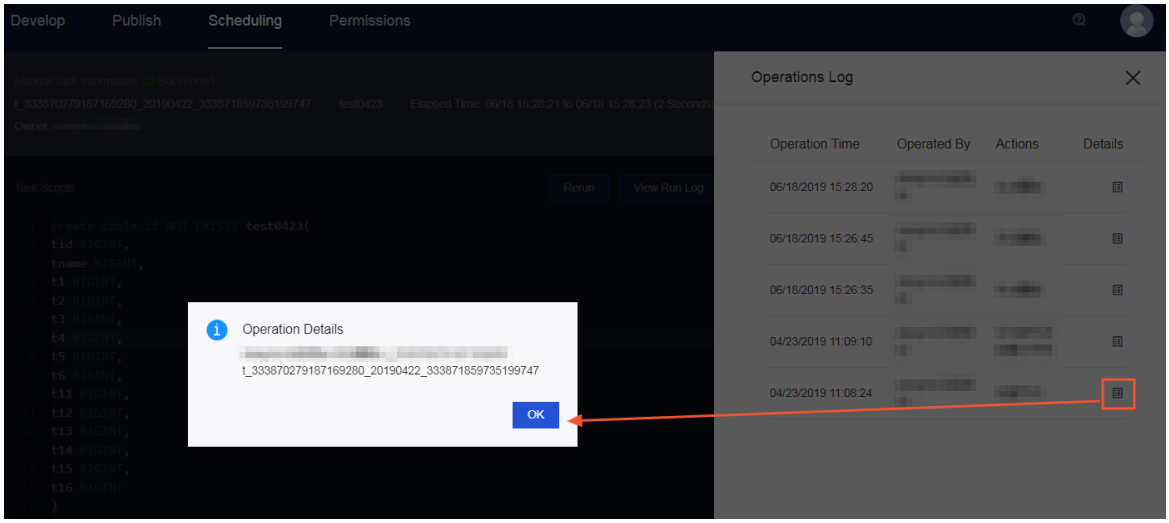
1. Click **View Operations Logs** at the top of the workspace, as shown in the following figure.



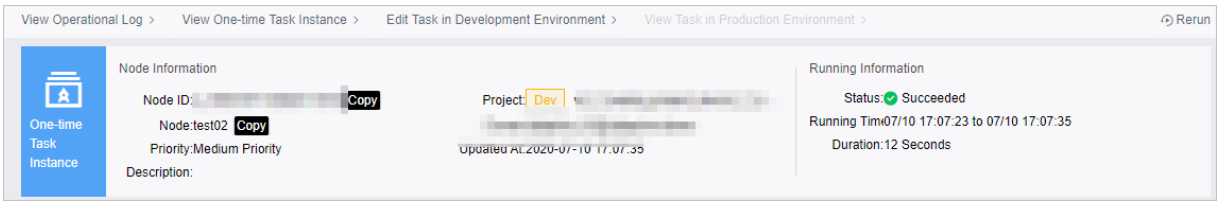
2. In the Operations Log dialog box that appears, you can view the operation history of the current task, as shown in the following figure.



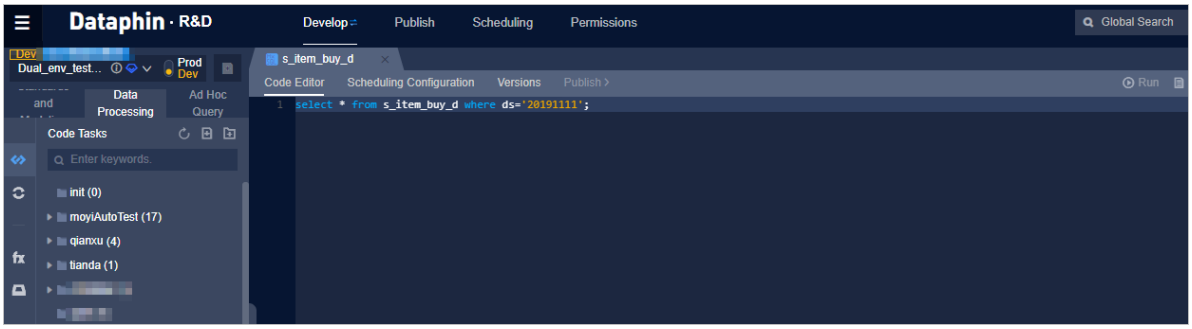
3. Click the icon in the **Details** column of an operation, as shown in the following figure. In the dialog box that appears, you can view the details of the operation.



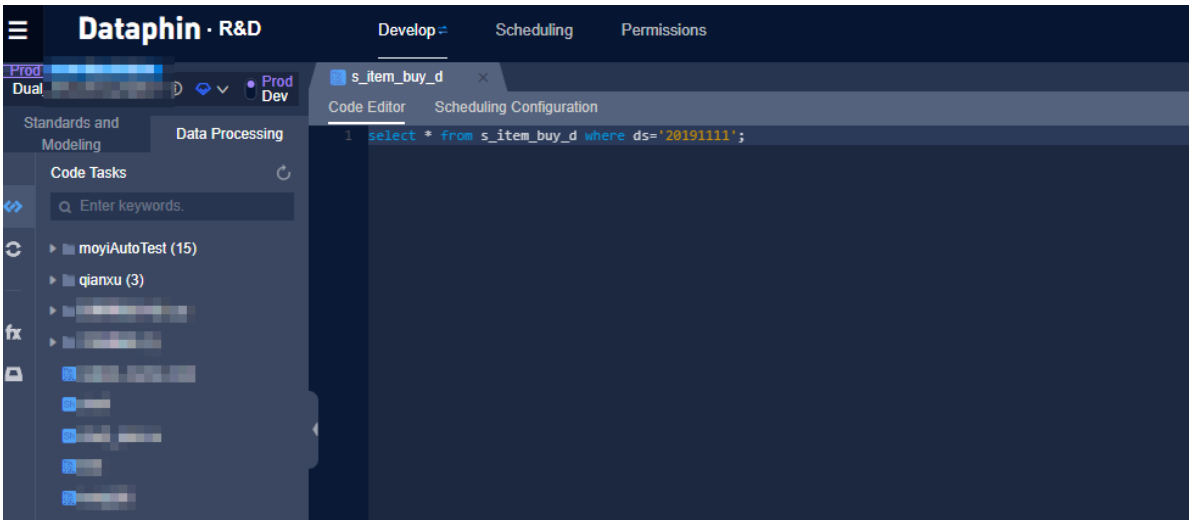
Edit a node



- Click **Edit Task in Development Environment** at the top of the workspace to go to the editing page of the node in the development environment, as shown in the following figure.



- Click **View Task in Production Environment** at the top of the workspace to go to the editing page of the node in the production environment, as shown in the following figure.



9.12.2.5. Recurring instances

Recurring instances are generated when recurring tasks are periodically scheduled in the production environment. This topic describes the operations that you can perform on recurring instances. For example, you can view recurring instances and use DAGs to manage these instances.

Go to the Recurring Task Instances section

To go to the Recurring Task Instances section, perform the following steps:

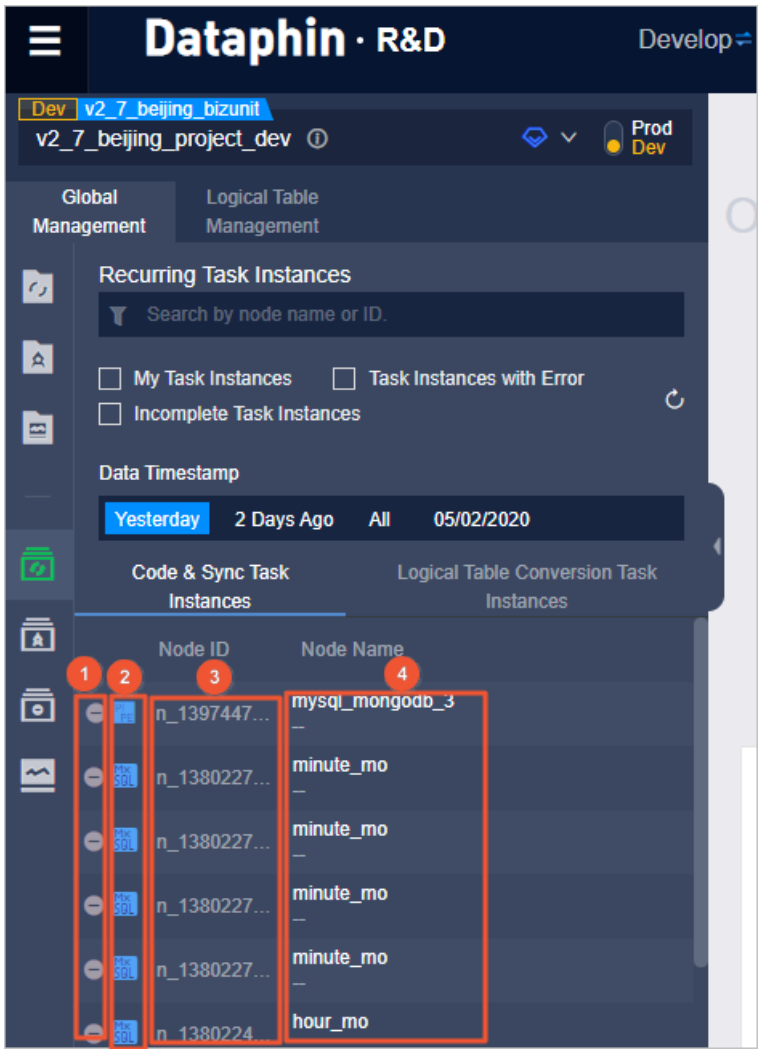
1. Log on to the Dataphin console.
2. Go to the **Global Management** tab.
    i. On the Dataphin homepage, click **R&D** in the top navigation bar.

ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.

iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

- If the project that you accessed last time is in **Basic** or **Prod** mode, use one of the following methods to go to the **Global Management** tab:
  - a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner.
    b. On the Scheduling page, click the icon next to the project name in the upper-left corner, click the **Dev** tab, and then select a project.
    c. The **Global Management** tab appears by default.
  - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
    b. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev** tab, and then select a project.
    c. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.
- If the project that you accessed last time is in **Dev** mode, use one of the following methods to go to the **Global Management** tab:
  - On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner. The **Global Management** tab appears by default.
  - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
    b. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

Click the icon on the left-side navigation submenu. The **Recurring Task Instances** section appears.



No.	Description
1	The running status of the instance.
2	The task type of the instance. Each task type is indicated by a unique icon. • : MAX_COMPUTE_SQL • : MAX_COMPUTE_MR • : SPARK_JAR_ON_MAX_COMPUTE • : SHELL • : PYTHON • : VIRTUAL

No.	Description
3	The node ID of the instance.
4	The node name of the instance.

Search for recurring instances

You can use one of the following methods to search for recurring instances:

- Click the icon in the search box, click the icon next to **Owner**, **Task Type**, or **Status**, and then select an owner, a task type, or a status. You can also specify a flexible combination of **Owner**, **Task Type**, and **Status** to search for your desired recurring instances.
- Enter an instance name in the search box to search for the specific instance. Alternatively, enter a keyword in the search box to search for instances whose names contain the keyword.

Manage a recurring instance in the right-side workspace

Right-click an instance node in the DAG. On the shortcut menu that appears, you can select menu items to perform supported operations.

- The following figure shows the shortcut menu for a recurring instance that is generated for a recurring task submitted or published from the **Integrated** module.

Menu item	Description
Show Parent Nodes	Select **Show Parent Nodes** and select the layer that you want to view.
Show Child Nodes	Select **Show Child Nodes** and select the layer that you want to view.
View Operational Log	Select **View Operational Log**. On the Operational Log page, view the operational logs of the instance.
View Node Script	Select **View Node Script**. On the Node Script page, view the code of the instance.
View Script	Select **View Script**. On the **Pipeline Details** page, view the pipeline script of the instance.
View Operations Log	Select **View Operations Log**. In the **Operations Log** pane, view the operation details of the instance.
Terminate	Select **Terminate** to terminate the instance if it is in the **Wait Submission** or **Running** state.
Rerun and Resume Scheduling	Select **Rerun and Resume Scheduling** to rerun the instance if it is in the **Success**, **Not Running**, or **Failed** state.
Rerun Downstream Nodes	Select **Rerun Downstream Nodes** to rerun the downstream nodes of the instance.
Set to Succeeded and Resume Scheduling	Select **Set to Succeeded and Resume Scheduling** to set the running status of the instance to Success and schedule the instance.

Menu item	Description
Remove Upstream Dependency	Select **Remove Upstream Dependency** to remove the upstream dependencies of the instance.
Force Rerun	Select **Force Rerun** to forcibly rerun the instance.
Pause	Select **Pause** to pause the instance.

- The following figure shows the shortcut menu for a recurring instance that is generated for a recurring task submitted or published from the **Develop** module.
  - Click **Edit Node in Development Environment** to edit the task corresponding to the instance in the development environment.
  - Click **View Node in Production Environment** to view the task corresponding to the instance in the production environment.

  For more information about other supported operations, see the preceding table.

9.12.2.6. Stream processing instances

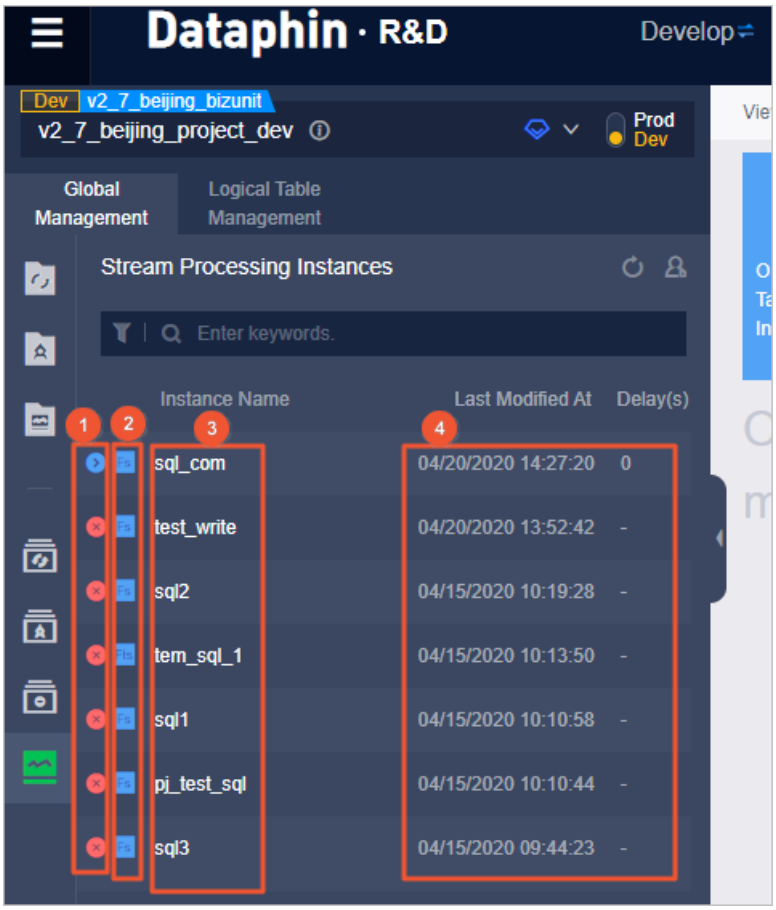
When you run a stream processing task in Dataphin, an instance is generated. A stream processing task has only one instance in an environment. This topic describes the operations that you can perform on stream processing instances.

Go to the Stream Processing Instances section

To go to the Stream Processing Instances section, perform the following steps:

1. Log on to the Dataphin console
2. Go to the **Global Management** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.
   iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

Click the icon on the left-side navigation submenu. The **Stream Processing Instances** section appears.



No.	Description
1	The running status of the instance. Each running status is indicated by a unique icon. • : Running • : Terminate • : Terminating • : Suspend • : Suspending • : Starting • : Failed to Run • After you resume a paused instance, the instance status changes from **Suspend** to **Resuming** and the icon changes to .

No.	Description
2	The task type of the instance. Each task type is indicated by a unique icon. • : FLINK_SQL • : FLINK_TEMPLATE_SQL
3	The name of the instance. The instance name is the name of the submitted stream processing task.
4	The time when the instance was last modified. By default, stream processing instances are sorted in descending order of running time.

Search for stream processing instances

- In the Stream Processing Instances section, click the icon and then the icon. In the drop-down list that appears, select **FLINK_SQL** or **FLINK_TEMPLATE_SQL** to filter instances of the specified type.
- Enter an instance name in the search box to search for the specific instance. Alternatively, enter a keyword in the search box to search for instances whose names contain the keyword.

View stream processing instances

Click the icon next to Stream Processing Instances to view the stream processing instances that you own.

Manage a stream processing instance in the right-side workspace

Click a stream processing instance in the **Stream Processing Instances** section. The DAG of the instance appears in the right-side workspace. The DAG shows the relationships between the nodes of the stream processing instance and the resource consumption of the nodes. After you click a node in the DAG, the node details appear in the lower-left corner of the workspace.

You can perform the following operations on the instance in the workspace:

- Click **Instance Code** in the top navigation bar of the workspace. In the Instance Code pane, view the code of the instance. You can also click **Go to R&D Workbench and Edit Task Code** to edit the code.
- Click **Instance Parameters** in the top navigation bar of the workspace. In the Instance Parameters pane, view the parameters of the instance. You can also click **Go to R&D Workbench and Edit Task Parameters** to modify the parameters.
- Click **Operations Log** in the top navigation bar of the workspace. In the Operations Log pane, view the operation logs of the instance.
- Click **View Stream Processing Task** in the top navigation bar of the workspace. In the Stream Processing Tasks section, view the stream processing task for which the instance was generated.
- If the task type is **FLINK_TEMPLATE_SQL**, click **Template Details** in the top navigation bar of the workspace. In the Template Details pane, view the template information. You can also click **Go to R&D Workbench to Edit Template** to edit the template information.
- Click **Runtime Analysis** in the top navigation bar of the workspace. On the page that appears,

view the running details of the instance, such as the failover information and logs.

In projects in Basic or Prod modes, you can pause, terminate, resume, rerun, and unpublish production instances. In projects in Dev mode, you can pause, rerun, and unpublish test instances. For more information, see the following description:

- For an instance in the **Running** state, you can click **Suspend** or **Terminate** in the upper-right corner of the workspace to pause or terminate the instance.
- For an instance in the **Suspend** state, you can click **Resume** or **Terminate** in the upper-right corner of the workspace to resume or terminate the instance.
- For an instance in the **Failed to Run** state, you can click **Rerun** or **Unpublish** in the upper-right corner of the workspace to rerun or delete the instance.

9.12.2.7. Retroactive data generation instances

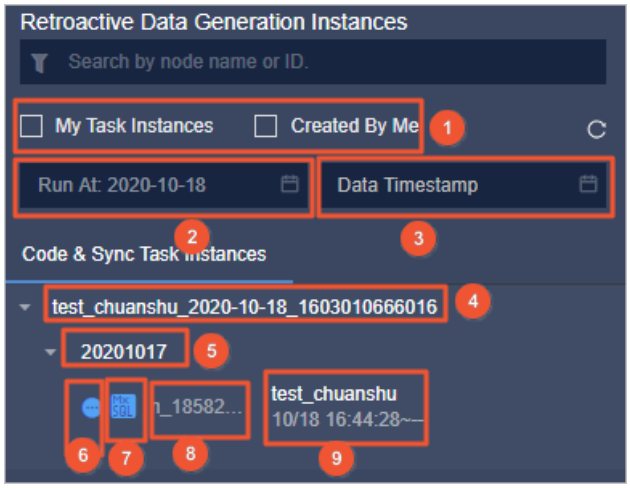
Retroactive data generation instances are generated when you generate retroactive data for recurring tasks. This topic describes the operations that you can perform on retroactive data generation instances.

Go to the Retroactive Data Generation Instances section

To go to the Retroactive Data Generation Instances section, perform the following steps:

1. Log on to the Dataphin console.
2. Go to the **Global Management** tab.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.
   iii. Click **Scheduling** in the top navigation bar. The **Global Management** tab appears by default.

Click the icon in the left-side navigation submenu. The **Retroactive Data Generation Instances** section appears. By default, this section displays the retroactive data generation instances that were run on the current day in descending order of running time.



No.	Description
1	The conditions for filtering the retroactive data generation instances that you own and those you created. You can select **My Task Instances** to filter the instances that you own or **Created By Me** to filter the instances that you created.
2	The date on which the retroactive data generation instance was run. You can click **Run At** and select a date to filter the instances that were run on that day.
3	The data timestamp of the retroactive data generation instance. You can click **Data Timestamp** and select a date to filter the instances that generate data for that date.
4	The display name of the retroactive data generation instance. The display name is in the format `Name of the task corresponding to the instance_Date on which the instance was run_Name of the instance`.
5	The data timestamp of the retroactive data generation instance. The data timestamp is the date for which the retroactive data generation instance generates data required by the corresponding recurring task.
6	The running status of the retroactive data generation instance.
7	The type of the recurring task for which the retroactive data generation instance was generated.
8	The node ID of the retroactive data generation instance.
9	The node name of the retroactive data generation instance.

Search for retroactive data generation instances

You can use one of the following methods to search for retroactive data generation instances:

- Click the icon in the search box, click the icon next to **Owner** or **Status**, and then select an owner or a status. You can also select both an owner and a status to search for your desired retroactive data generation instances.
- Enter the name of a retroactive data generation instance in the search box to search for the specific instance. Alternatively, enter a keyword in the search box to search for retroactive data generation instances whose names contain the keyword.

Manage a retroactive data generation instance in the right-side workspace

Click a retroactive data generation instance in the **Retroactive Data Generation Instances** section. The DAG of the instance appears in the right-side workspace. In the DAG, you can perform the following operations:

- Right-click the root node. On the shortcut menu that appears, select menu items to perform supported operations.

Menu item	Description
Show Parent Nodes	Select Show Parent Nodes and select the layer that you want to view.
Show Child Nodes	Select Show Child Nodes and select the layer that you want to view.
View Operational Log	Select **View Operational Log**. On the Operational Log page, view the operational logs of the root node.

- Right-click the instance node. On the shortcut menu that appears, select menu items to perform supported operations.

Menu item	Description
Show Parent Nodes	Select **Show Parent Nodes** and select the layer that you want to view.
Show Child Nodes	Select **Show Child Nodes** and select the layer that you want to view.
View Operational Log	Select **View Operational Log**. On the Operational Log page, view the operational logs of the instance.
View Node Script	Select **View Node Script**. On the Node Script page, view the code of the task corresponding to the instance in the development environment.
Edit Node in Development Environment	Select **Edit Task in Development Environment** to edit the task corresponding to the instance in the development environment.
View Node in Production Environment	Select **View Task in Production Environment** to view the code of the task corresponding to the instance in the production environment.
View Operations Log	Select **View Operations Log**. In the **Operations Log** pane, view the operation details of the instance.
Terminate	Select **Terminate** to terminate the instance if it is in the **Wait Submission** or **Running** state.
Rerun and Resume Scheduling	Select **Rerun and Resume Scheduling** to rerun the instance if it is in the **Success**, **Not Running**, or **Failed** state.
Rerun Downstream Nodes	Select **Rerun Downstream Nodes** to rerun the downstream nodes of the instance.
Set to Succeeded and Resume Scheduling	Select **Set to Succeeded and Resume Scheduling** to set the running status of the instance to Success and schedule the instance.
Remove Upstream Dependency	Select **Remove Upstream Dependency** to remove the upstream dependencies of the instance.
Force Rerun	Click **Force Rerun** to forcibly rerun the instance.
Pause	Click **Pause** to pause the instance.

9.12.3. Logical tables

9.12.3.1. Logical table tasks

This topic describes the operations that you can perform on logical table tasks.

Go to the Logical Table Tasks section

1. Log on to the Dataphin console.
2. Go to the **Logical Table Tasks** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.
   iii. Click **Scheduling** in the top navigation bar. The **Scheduling** page appears.
   iv. In the left-side navigation pane, click the **Logical Table Management** tab. The **Logical Table Tasks** section appears by default.
3. In the **Logical Table Tasks** section, you can view and search for logical tables and their conversion tasks. If you click a logical table in the Logical Table Tasks section, the DAG of the logical table appears in the right-side workspace. The DAG shows all the task nodes of the logical table and their relationships. The thumbnail of the DAG appears in the lower-right corner of the workspace.

No.	Description
1	The logical table.
2	The logical table tasks.

View the fields that are contained in a logical table task

1. In the **Logical Table Tasks** section, click the target logical table.
2. Move the pointer over the target logical table task and click the icon.
3. In the Logical Table Conversion Task Node pane, view the fields that are contained in the logical table task.

View the code of a logical table task

1. In the **Logical Table Tasks** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task and select **View Node Script**.
2. On the **Node Script** page, view the code of the logical table task.

View the global DAG of a logical table task

1. In the **Logical Table Tasks** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task and select **View Recurring Task**.
2. On the **Global Management** tab, view the logical table task in the **Recurring Tasks** section and the global DAG of the task in the right-side workspace.

View the global instance DAG of a logical table task

1. In the **Logical Table Tasks** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task and choose **View Instances** > **View Recurring Instances**.
2. On the **Global Management** tab, view the instance of the logical table task in the **Recurring Task Instances** section and the global DAG of the instance in the right-side workspace.

View the retroactive data generation instances that have been generated for a logical table task

1. In the **Logical Table Tasks** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task and choose **View Instances** > **View Retroactive Data Generation Instances**.
2. On the **Global Management** tab, view the retroactive data generation instances that have been generated for the logical table task in the **Retroactive Data Generation Instances** section.

Generate retroactive data for a logical table task

1. In the **Logical Table Tasks** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task and select **Generate Retroactive Data**.
2. In the **Generate Retroactive Data** dialog box, set the **Instance Name**, **Data Timestamp**, and **Select Downstream Nodes** parameters.

   If you set **Select Downstream Nodes** to **Yes**, you must select downstream nodes for which you want to generate retroactive data.
3. Click **OK**.

9.12.3.2. Logical table task instances

Logical table task instances are generated when logical table tasks are run. This topic describes the operations that you can perform on logical table task instances.

Go to the Logical Table Task Instances section

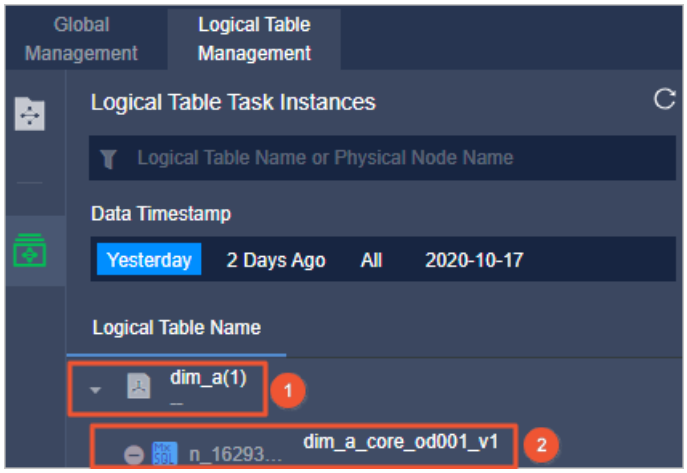
1. Log on to the Dataphin console.
2. Go to the **Logical Table Task Instances** section.
   i. On the Dataphin homepage, click **R&D** in the top navigation bar.
   ii. On the Develop page, click the icon next to the project name in the upper-left corner, click the **Dev**, **Basic**, or **Prod** tab, and then select a project.

iii. Click **Scheduling** in the top navigation bar. The **Scheduling** page appears.

iv. In the left-side navigation pane, click the **Logical Table Management** tab. The **Logical Table Tasks** section appears by default.

v. On the left-side navigation submenu, click the icon. The **Logical Table Task Instances** section appears.

3. In the **Logical Table Task Instances** section, you can view and search for logical tables and their conversion task instances. If you click a logical table in the Logical Table Task Instances section, the DAG of the logical table appears in the right-side workspace. The DAG shows all the task instance nodes of the logical table and their running status, for example, Running, Success, or Failed. The thumbnail of the DAG appears in the lower-right corner of the workspace.Logical table task instances are displayed in a two-level hierarchy.



No.	Description
1	The logical table.
2	The logical table task instances.

View the fields that are contained in a logical table task instance

1. In the **Logical Table Task Instances** section, click the target logical table.
2. Move the pointer over the target logical table task instance and click the icon.
3. In the Logical Table Conversion Task Node pane, view the fields that are contained in the logical table task instance.

View the operational logs of a logical table task instance

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and select **View Operational Log**.
2. On the **Operational Log** page, view the operational logs of the logical table task instance.

View the code of a logical table task instance

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and

select **View Node Script**.

2. On the **Node Script** page, view the code of the logical table task instance.

Rerun a logical table task instance

> **Note** You can rerun a logical table task instance only when the instance is in the **Success**, **Not Running**, or **Failed** state.

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and select **Rerun and Resume Scheduling**.
2. In the **Rerun and Resume Scheduling** message, click **OK**.

View the global DAG of the logical table task for which a logical table task instance was generated

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and select **View Recurring Task**.
2. On the **Global Management** tab, view the task for which the logical table task instance was generated in the **Recurring Tasks** section and the global DAG of the task in the right-side workspace.

View the global DAG of a logical table task instance

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and choose **View Instances** > **View Recurring Instances**.
2. On the **Global Management** tab, view the logical table task instance in the Recurring Task Instances section and the global DAG of the instance in the right-side workspace.

View the retroactive data generation instances that have been generated for the task corresponding to a logical table task instance

1. In the **Logical Table Task Instances** section, click the target logical table. In the DAG in the right-side workspace, right-click the node of the target logical table task instance and choose **View Instances** > **View Retroactive Data Generation Instances**.
2. On the **Global Management** tab, view the retroactive data generation instances that have been generated for the task corresponding to the logical table task instance in the **Retroactive Data Generation Instances** section.

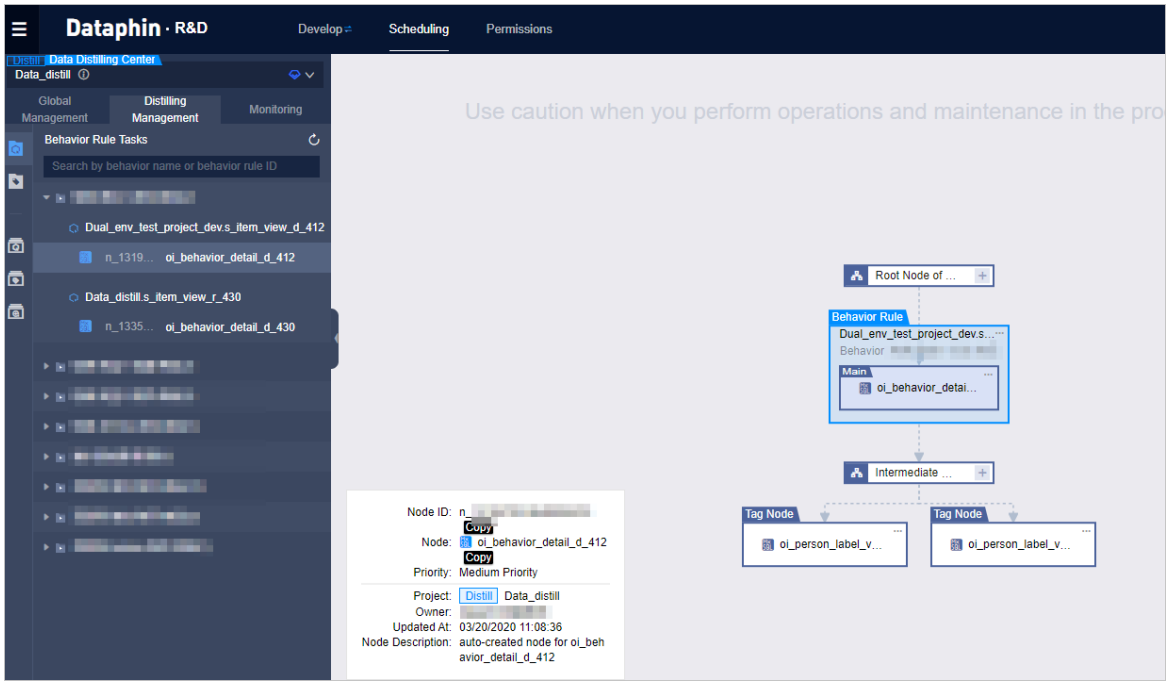
9.12.4. Distilling management

9.12.4.1. Behavior rule tasks

This topic describes the operations that you can perform on behavior rule tasks.

Behavior Rule Tasks page

1. Go to the scheduling center. Click the icon at the top of the left-side navigation pane and select the Data_distill project on the **Basic** tab. The **Distilling Management** tab appears.

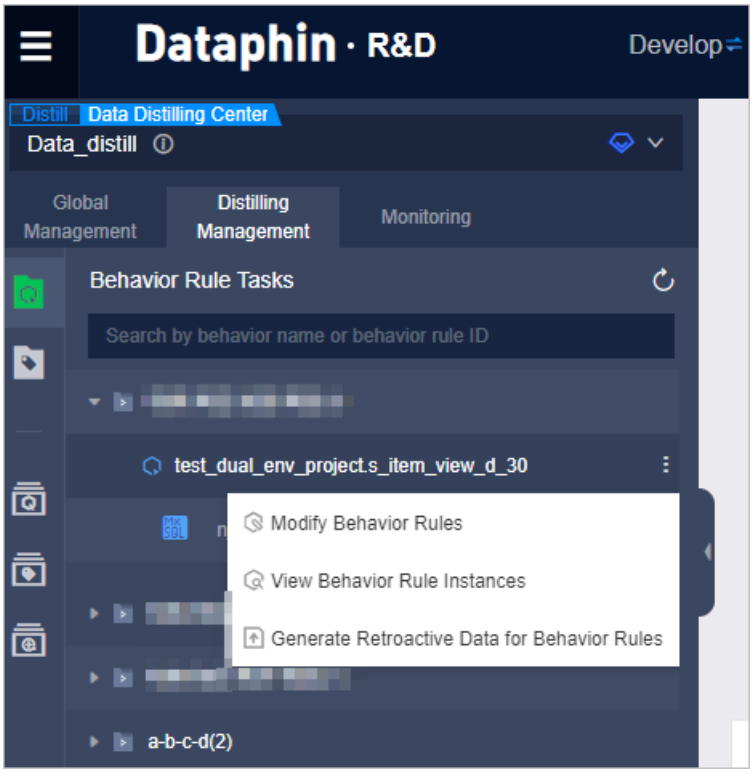


2. On the left-side navigation submenu, click **Behavior Rule Tasks**. The **Behavior Rule Tasks** page appears. In the left-side navigation pane, the behavior rule tasks are sorted in descending order by update time, and the first behavior rule task is selected by default.
   - In the left-side navigation pane, you can view and search for behavior rules and their nodes. You can enter a behavior name or behavior rule ID in the search box in the upper-left corner to search for a behavior rule task. When you search for a behavior rule task, the behavior rule ID must be exactly matched, whereas the behavior name supports fuzzy match.

     In the left-side navigation pane, the behavior rule tasks are displayed in three layers. The first layer displays the behavior. The second layer displays the behavior rules under the behavior. The third layer displays the task node of the behavior rule, including the type (displayed as an icon), ID, and name of the node. Move the pointer over a node. The ID, name, owner, and update time of the node appear.
   - The directed acyclic graph (DAG) in the workspace shows the node of the current behavior rule, the common nodes of the upstream data distilling center related to the current behavior rule, and the nodes of the downstream tags. The **Behavior Rule** box is added outside the node of the current behavior rule to display the behavior rule information. By default, all nodes except the node of the current behavior rule and the nodes of the downstream tags are hidden in node groups. You can click the **Expand** icon of a node group to view the nodes in the node group and their relationships in the DAG. In addition, the thumbnail of the DAG is available in the lower-right corner of the workspace.

Manage behavior rule tasks

- In the left-side navigation pane, move the pointer over the icon of a behavior. You can generate retroactive data for the behavior rules under the behavior.



Select **Generate Retroactive Data**. In the **Generate Retroactive Data for Behavior Rules** dialog box that appears, set the parameters as prompted and click **OK**.

- In the left-side navigation pane, move the pointer over the icon of a behavior rule. You can edit the behavior rule, view the task instances of the behavior rule, or generate retroactive data for the behavior rule.
  - Select **Modify Behavior Rules**. The **Change Behavior Rules** tab appears. For more information about how to edit the behavior rule, see Edit a behavior rule.
  - Select **View Behavior Rule Instances**. On the **Behavior Rule Task Instances** page that appears, you can view the task instances of the behavior rule. The default data timestamp is yesterday.
  - Select **Generate Retroactive Data for Behavior Rules**. In the **Generate Retroactive Data for Behavior Rules** dialog box that appears, set the parameters as prompted and click **OK**.

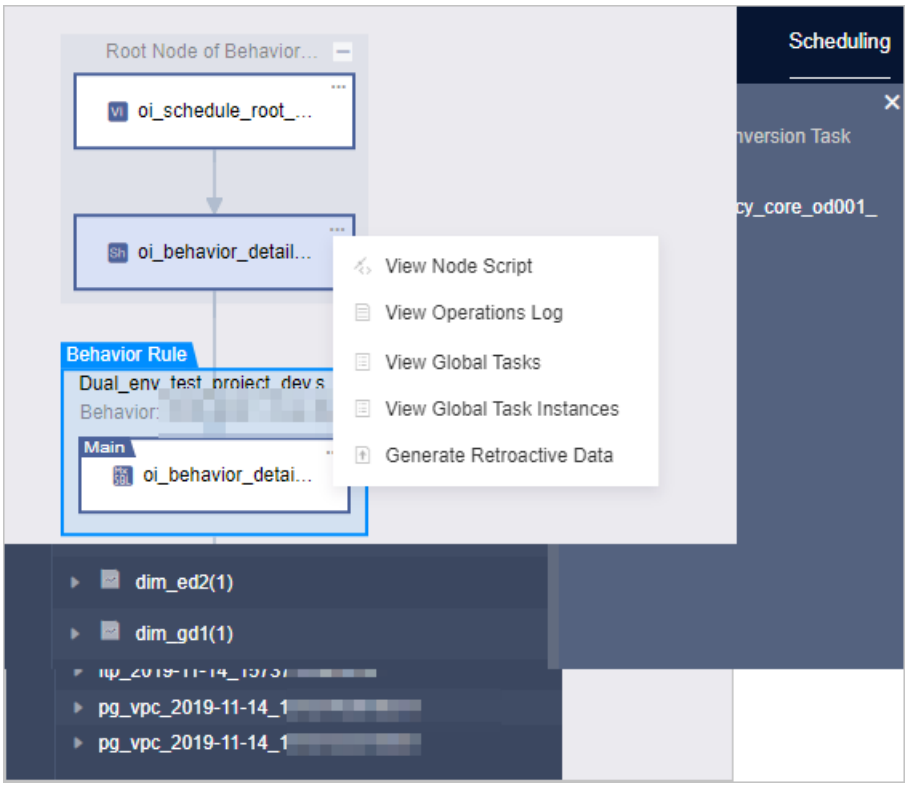
Shortcut operation items

In the upper-right corner of the workspace, the **Show All Node Groups**, **Hide All Node Groups**, and **Fit to Canvas** icons are available. In the upper-right corner of the workspace, you can click the **Refresh** button to view the latest node relationships. You can also search for a node in the DAG by node ID or output field name. A toolbar is available at the bottom of the workspace. You can zoom in or zoom out the DAG and view the DAG in full screen. You can drag the DAG or click a node to view the detailed information about the node in the lower-left corner of the workspace.

- Click the icon of the **Behavior Rule** box or right-click the **Behavior Rule** box. You can generate retroactive data for the behavior rule, edit the behavior rule, view behavior rule task instances, or view retroactive data generation instances.
  - Select **Generate Retroactive Data for Behavior Rules**. In the **Generate Retroactive Data for Behavior Rules** dialog box that appears, set the parameters as prompted and click **OK**.

- Select **Modify Behavior Rules**. On the **Change Behavior Rules** tab that appears, set the parameters as prompted. For more information, see Edit a behavior rule.
- Choose **View Instance** > **View Behavior Rule Instances**. On the **Behavior Rule Task Instances** page that appears, you can view the task instances of the behavior rule. The default data timestamp is yesterday.
- Choose **View Instance** > **View Retroactive Data Generation Instance of Behavior Rule**. On the **Retroactive Data Generation Instances** page that appears, you can view the retroactive data generation instances of the behavior rule. The default data timestamp is today.

- Click the **More** icon of a node (including the node of the current behavior rule) or right-click the node. You can view the node script, view operations logs, view global tasks, view global task instances, or generate retroactive data.



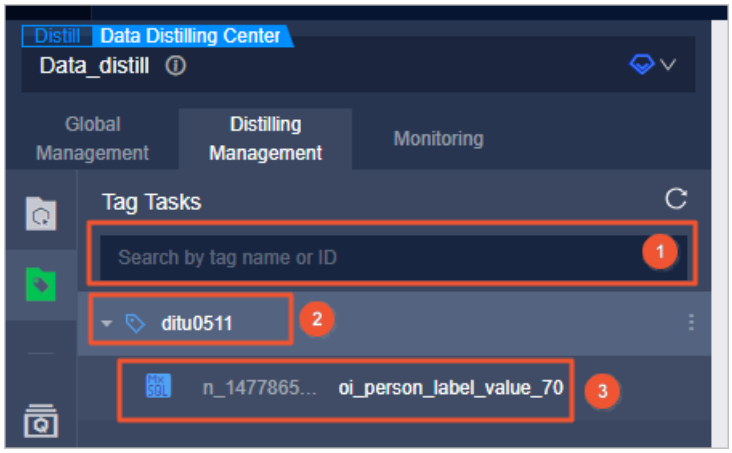
- Select **View Node Script**. On the **Node Script** page that appears, you can view the script of the node, but you cannot edit it.
- Select **View Operations Log**. In the **Operations Log** dialog box that appears, you can view the operation history of the node.
- Select **View Global Tasks**. On the **Recurring Tasks** page that appears, you can view the global DAG of the node.
- Select **View Global Task Instances**. On the **Recurring Task Instances** page that appears, you can view the global DAG of the node.
- Select **Generate Retroactive Data**. In the **Generate Retroactive Data** dialog box that appears, set the parameters as prompted and click **OK**.

9.12.4.2. Tag tasks

This topic describes the operations that you can perform on tags and tag tasks. For example, you can edit a tag, generate retroactive data for a tag, and view the task instances of a tag.

Go to the Tag Tasks section

1. Log on to the Dataphin console.
2. Go to the **Distilling Management** tab.
   - If the project that you accessed last time is the **Data_distill** project, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner of the Dataphin homepage.
   - If the project that you accessed last time is not the **Data_distill** project, use one of the following methods to go to the **Distilling Management** tab:
     - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
       b. On the **Develop** page, move the pointer over **Develop** in the top navigation bar and select **Distilling**.
       c. On the **Distilling** page, click **Scheduling** in the top navigation bar.
     - a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner. The **Scheduling** page appears.
       b. Click the icon next to the project name in the upper-left corner, click the **Basic** tab, and then select the **Data_distill** project.
3. On the left-side navigation submenu, click the icon.
4. In the **Tag Tasks** section, view tag tasks.



No.	Description
1	The search box for you to search for tag tasks by tag name or ID.
2	The tag.
3	The tag task. The display name of a tag task is in the format of Tag task type_Tag task ID_Tag task name.

Generate retroactive data for a tag

1. Open the **Generate Retroactive Data for Tags** dialog box by using one of the following methods:

- In the **Tag Tasks** section, move the pointer over the icon next to the target tag and select **Generate Retroactive Data for Tags**.
- In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the Tag box and select **Generate Retroactive Data for Tags**.

2. In the **Generate Retroactive Data for Tags** dialog box, set the parameters for the retroactive data generation instance.

Section	Operation
Effective Period	Specify the end date of the period for which you want to generate retroactive data.
Instance Name	Set the name of the retroactive data generation instance to be generated for the tag.

3. Click **OK**.

Edit a tag

1. Go to the Change Tag tab by using one of the following methods:
   - In the **Tag Tasks** section, move the pointer over the icon next to the target tag and select **Edit Tag**.
   - In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the Tag box and select **Edit Tag**.
2. On the Change Tag tab, modify the tag parameters as required. For more information, see Edit a factory tag.

View the task instances of a tag

1. Go to the **Tag Task Instances** section by using one of the following methods:
   - In the **Tag Tasks** section, move the pointer over the icon next to the target tag and select **View Tag Instances**.
   - In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the Tag box and choose **View Instance** > **Tag Instance**.
2. In the **Tag Task Instances** section, view the task instances that have been generated for the tag.

View the retroactive data generation instances for a tag

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the Tag box and choose **View Instance** > **Retroactive Data Generation Instance for Tags**.
2. In the **Retroactive Data Generation Instances** section, view the retroactive data generation instances that have been generated for the tag. You can view the details of each retroactive data generation instance.

View the code of a tag task

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the task node and select **View Node Script**.
2. On the **Node Script** page, view the code of the task.

View the operation logs of a tag task

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the task node and select **View Operations Log**.
2. In the **Operations Log** pane, view the operation logs of the task. You can click the icon in the **Details** column to view the operation details.

View the global DAG of a tag task

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the task node and select **View Global Tasks**.
2. On the **Global Management** tab, view the tag task in the **Recurring Tasks** section and the global DAG of the task in the right-side workspace.

View the global instance DAG of a tag task

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the task node and select **View Global Task Instances**.
2. On the **Global Management** tab, view the instance of the tag task in the **Recurring Task Instances** section and the global DAG of the instance in the right-side workspace.

Generate retroactive data for a tag task

1. In the **Tag Tasks** section, click the target tag and then the target tag task. In the DAG in the right-side workspace, right-click the task node and select **Generate Retroactive Data**.
2. In the **Generate Retroactive Data** dialog box, set the **Instance Name**, **Data Timestamp**, and **Select Downstream Nodes** parameters.

   If you set **Select Downstream Nodes** to **Yes**, you must select downstream nodes for which you want to generate retroactive data.
3. Click **OK**.

9.12.4.3. Behavior rule task instances

Behavior rule task instances are generated when behavior rule tasks are run. This topic describes the operations that you can perform on behavior rule task instances.

Go to the Behavior Rule Task Instances section

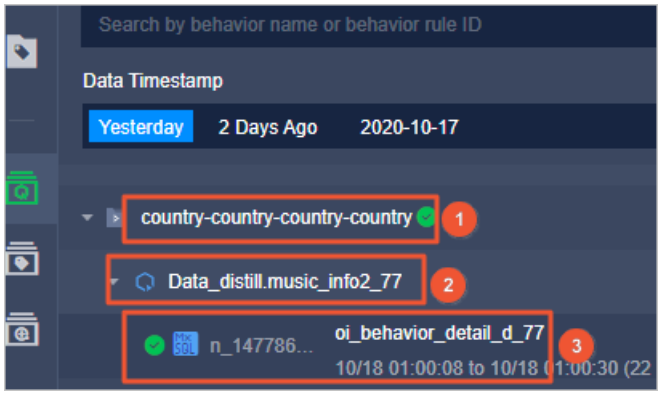
1. Log on to the Dataphin console.
2. Go to the **Distilling Management** tab.
   - If the project that you accessed last time is the **Data_distill** project, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner of the Dataphin homepage.
   - If the project that you accessed last time is not the **Data_distill** project, use one of the

following methods to go to the **Distilling Management** tab:

- a. On the Dataphin homepage, click **R&D** in the top navigation bar.
  b. On the **Develop** page, move the pointer over **Develop** in the top navigation bar and select **Distilling**.
  c. On the **Distilling** page, click **Scheduling** in the top navigation bar.
- a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner. The **Scheduling** page appears.
  b. Click the icon next to the project name in the upper-left corner, click the **Basic** tab, and then select the **Data_distill** project.

3. On the left-side navigation submenu, click the icon.
4. In the **Behavior Rule Task Instances** section, view behavior rule task instances.

   You can view and search for behaviors, behavior rules, and behavior rule task instances in the Behavior Rule Task Instances section. If you click a behavior rule task instance in the Behavior Rule Task Instances section, the DAG of the instance appears in the right-side workspace. The DAG shows the node of the current behavior rule task instance and its upstream and downstream nodes. The thumbnail of the DAG appears in the lower-right corner of the workspace.Behavior rule task instances are displayed in a three-level hierarchy.



No.	Description
1	The behavior and the overall running status of all task instances under the behavior. ○ The overall running status is Success only if all task instances under the behavior are properly run. ○ The overall running status is Failed if any task instance under the behavior failed. ○ The overall running status is Running if some task instances under the behavior are properly run and the others are running.
2	The behavior rule.

No.	Description
3	The behavior rule task instances. You can move the pointer over an instance to view the instance information, including the instance ID, name, owner, and running time.

Search for behavior rule task instances

In the **Behavior Rule Task Instances** section, you can search for behavior rule task instances by using one of the following methods:

- Click **Yesterday** or **2 Days Ago** below **Data Timestamp** to filter behavior rule task instances.
- Enter a behavior name or behavior rule ID in the search box to search for specific behavior rule task instances.
- Click the date below Data Timestamp and select a date to filter behavior rule task instances.

Modify a behavior rule

1. Go to the Change Behavior Rules tab by using one of the following methods:
   - In the **Behavior Rule Task Instances** section, click the target behavior. Move the pointer over the icon next to the target behavior rule and select **Modify Behavior Rules**.
   - In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the Behavior Rule box and select **Modify Behavior Rules**.
2. On the **Change Behavior Rules** tab, modify the behavior rule. For more information, see Edit a behavior rule.

View the behavior rule task under a behavior rule

1. Go to the **Behavior Rule Tasks** section by using one of the following methods:
   - In the **Behavior Rule Task Instances** section, click the target behavior. Move the pointer over the icon next to the target behavior rule and select **View Behavior Rule Tasks**.
   - In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the Behavior Rule box and select **View Behavior Rule Tasks**.
2. In the **Behavior Rule Tasks** section, view the behavior rule task under the behavior rule. You can view the details of the task.

View the operational logs of a behavior rule task instance

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **View Operational Log**.
2. On the **Operational Log** page, view the operational logs of the instance.

View the code of a behavior rule task instance

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **View Node Script**.

2. On the **Node Script** page, view the code of the instance.

View the operation logs of a behavior rule task instance

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **View Operations Log**.
2. In the **Operations Log** pane, view the operation logs of the instance. You can click the  icon in the **Details** column to view the operation details.

View the global DAG of the behavior rule task for which a behavior rule task instance was generated

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **View Global Tasks**.
2. On the **Global Management** tab, view the behavior rule task for which the instance was generated in the **Recurring Tasks** section and the global DAG of the task in the right-side workspace.

View the global DAG of a behavior rule task instance

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **View Global Task Instances**.
2. On the **Global Management** tab, view the instance in the **Recurring Task Instances** section and the global DAG of the instance in the right-side workspace.

Terminate a behavior rule task instance

> **Note** You can terminate a behavior rule task instance only when the instance is in the **Wait Submission** or **Running** state.

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **Terminate**.
2. In the **Terminate** message, click **OK**.

Rerun a behavior rule task instance

> **Note**
>
> - You can rerun a behavior rule task instance only when the instance is in the **Success**, **Not Running**, or **Failed** state.
> - Use caution when you rerun a behavior rule task instance, because the data generated by the instance previously will be overwritten after the instance is rerun.

1. In the **Behavior Rule Task Instances** section, click the target behavior, behavior rule, and behavior rule task instance in sequence. In the DAG in the right-side workspace, right-click the instance node and select **Rerun and Resume Scheduling**.

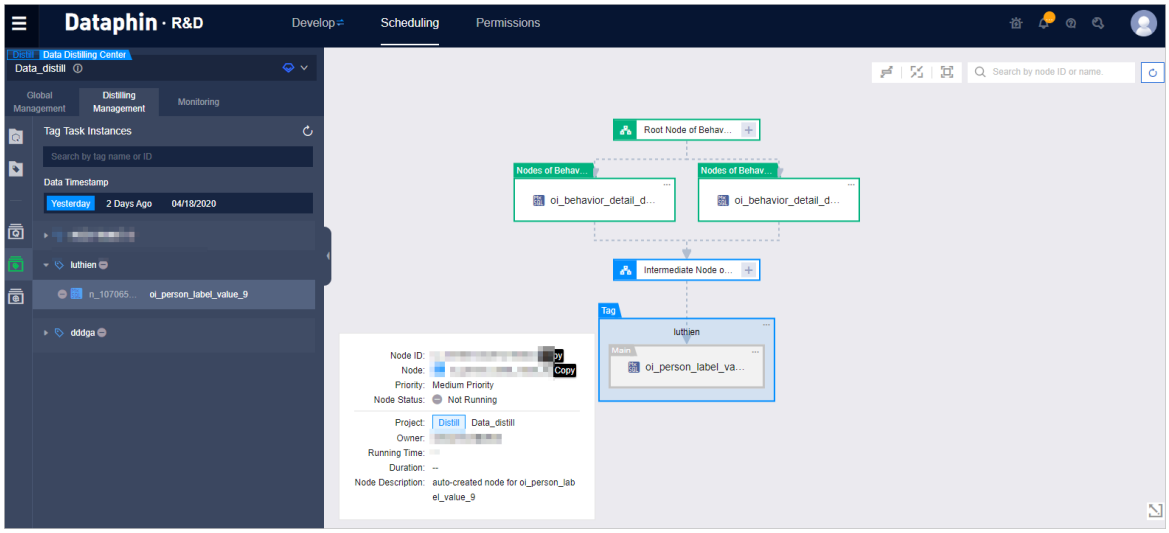
2. In the **Rerun and Resume Scheduling** message, click **OK**.

9.12.4.4. Tag task instances

This topic describes the operations that you can perform on tag task instances.

Tag Task Instances page

1. Go to the scheduling center. By default, the **Distilling Management** tab appears, as shown in the following figure.



2. On the left-side navigation submenu, click **Tag Task Instances**. The **Tag Task Instances** page appears.
   - In the left-side navigation pane, you can view and search for tags and their nodes. You can enter a tag name or tag ID in the search box in the upper-left corner to search for a tag task. When you search for a tag task, the tag ID must be exactly matched, whereas the tag name supports fuzzy match. The default data timestamp of the searched instances is yesterday. You can select **2 Days Ago** or a specific date to filter the search results.
   - In the left-side navigation pane, the tag task instances are displayed in two layers. The first layer displays the tag. The second layer displays the task instance node of the tag, including the instance status, type, ID, name, and running time of the node. Move the pointer over a node. The ID, name, owner, and running time of the node appear. In the left-side navigation pane, move the pointer over the icon of a tag to perform the following operations:
     - Select **Edit Tag**. On the **Change Tag** tab that appears, set the parameters as prompted. For more information, see Edit a factory tag.
     - Select **View Tag Tasks**. On the **Tag Tasks** page that appears, view the details of the task.
   - The color of a node box indicates the running status of the node. Green indicates that the node runs successfully. Red indicates that the node fails to run. Gray indicates that the node does not run. Blue indicates that the node is running. Purple indicates that the node is waiting to run. The color of a node group box indicates the running status of the node group. If all nodes in the node group run successfully, the node group appears in green. If

a node in the node group fails to run, the node group appears in red.

- The directed acyclic graph (DAG) in the workspace shows the node of the current tag, the intermediate nodes of related tags, the nodes of the behavior rules on which the current tag depends, and the common nodes of the upstream data distilling center related to the behavior rules. The **Tag** box is added outside the node of the current tag to display the tag information. By default, all nodes except the nodes of the behavior rules and the node of the current tag are hidden in node groups. You can click the Expand icon of a node group to view the nodes in the node group and their relationships in the DAG. In addition, the thumbnail of the DAG is available in the lower-right corner of the workspace.

Shortcut operation items

- Click the icon of the **Tag** box or right-click the **Tag** box to perform the following operations:
  - Select **Edit Tag**. On the **Change Tag** tab that appears, set the parameters as prompted. For more information, see Edit a factory tag.
  - Select **View Tag Tasks**. On the **Tag Tasks** page that appears, view the details of the task.
- Click the icon of a node (including the node of the current tag) or right-click the node to perform the following operations:
  - Select **View Operational Log**. On the **Operational Log** page that appears, you can view the operational logs of the node.
  - Select **View Node Script**. On the **Node Script** page that appears, you can view the script of the node, but you cannot edit it.
  - Select **View Operations Log**. In the **Operations Log** dialog box that appears, you can view the operation history of the node.
  - Select **View Global Tasks**. On the **Recurring Tasks** page that appears, you can view the global DAG of the node.
  - Select **View Global Task Instances**. On the **Recurring Task Instances** page that appears, you can view the global DAG of the node.
  - Select **Terminate** to terminate the node in the **Waiting** or **Running** state.
  - Select **Rerun and Resume Scheduling** to rerun the node in the **Success**, **Failed**, or **Not Running** state.

9.12.4.5. Retroactive data generation instances

The Retroactive Data Generation Instances section displays retroactive data generation instances that are generated when you generate retroactive data for tags and behavior rules. This topic describes the operations that you can perform on retroactive data generation instances.

Go to the Retroactive Data Generation Instances section

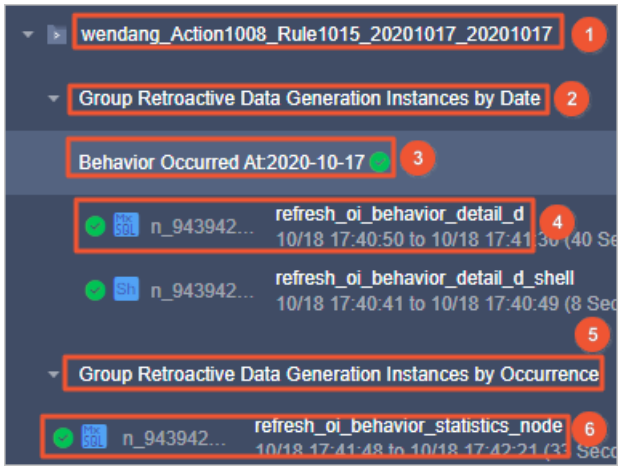
1. Log on to the Dataphin console.
2. Go to the **Distilling Management** tab.
   - If the project that you accessed last time is the **Data_distill** project, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner of the Dataphin homepage.

- If the project that you accessed last time is not the **Data_distill** project, use one of the following methods to go to the **Distilling Management** tab:
  - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
    b. On the **Develop** page, move the pointer over **Develop** in the top navigation bar and select **Distilling**.
    c. On the **Distilling** page, click **Scheduling** in the top navigation bar.
  - a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner. The **Scheduling** page appears.
    b. Click the icon next to the project name in the upper-left corner, click the **Basic** tab, and then select the **Data_distill** project.

3. On the left-side navigation submenu, click the icon.
4. In the **Retroactive Data Generation Instances** section, view the retroactive data generation instances that are generated for tags and behavior rules.

   You can also search for retroactive data generation instances in the Retroactive Data Generation Instances section. If you click a retroactive data generation instance for a behavior rule or a tag in the Retroactive Data Generation Instances section, the DAG of the instance appears in the right-side workspace. The DAG shows the node of the current retroactive data generation instance and its relationships with other retroactive data generation instances for the same behavior rule or tag. The thumbnail of the DAG appears in the lower-right corner of the workspace. The **Retroactive Data Generation Instances** section contains the **Retroactive Data Generation Instance for Behavior Rule** and **Retroactive Data Generation Instance for Tag** tabs. You can view different instance information on the two tabs.

   - The following table describes the instance information that you can view on the **Retroactive Data Generation Instance for Behavior Rule** tab.



No.	Description
1	The name of the folder for storing retroactive data generation instances for a behavior rule. The name is in the format of Prefix_Behavior ID_Behavior rule ID_Start date for generating retroactive data_End date for generating retroactive data.

No.	Description
2	A type of retroactive data generation instances. ■ **Group Retroactive Data Generation Instances by Date**: This type of instances are run every day in the specified period. You can specify a date in the Group Retroactive Data Generation Instances by Date box in the DAG to filter specific instances of this type. ■ **Group Retroactive Data Generation Instances by Occurrence**: This type of instances are run only once. Dataphin runs this type of instances for a behavior rule only after all instances of the **Group Retroactive Data Generation Instances by Date** type are properly run for the behavior rule.
3	The data timestamp and overall running status of retroactive data generation instances of the **Group Retroactive Data Generation Instances by Date** type. ■ The overall running status is Success only when all the instances are properly run. ■ The overall running status is Failed if any instance failed. ■ The overall running status is Running if some instances are properly run and the others are running.
4	The display name of a retroactive data generation instance of the **Group Retroactive Data Generation Instances by Date** type. The display name consists of the running status of the instance, type of the corresponding behavior rule task, ID of the instance, and name of the instance.
5	A type of retroactive data generation instances.
6	The display name of a retroactive data generation instance of the **Group Retroactive Data Generation Instances by Occurrence** type. The display name consists of the running status of the instance, type of the corresponding behavior rule task, ID of the instance, and name of the instance.

- **The following table describes the instance information that you can view on the Retroactive Data Generation Instance for Tag tab.**

No.	Description
1	The name of the folder for storing retroactive data generation instances for a tag. The name is in the format of Prefix_Tag ID_End date for generating retroactive data.
2	The display name of a retroactive data generation instance for a tag. The display name consists of the running status of the instance, type of the corresponding tag task, ID of the instance, and name of the instance.

Search for retroactive data generation instances

In the **Retroactive Data Generation Instances** section, retroactive data generation instances are sorted in descending order of generation time. You can search for retroactive data generation instances by using one of the following methods:

- Enter an instance name in the search box to search for specific retroactive data generation instances. On the Retroactive Data Generation Instance for Behavior Rule tab, the instances whose data timestamp is the current day are found by default. On the Retroactive Data Generation Instance for Tag tab, the instances whose data timestamp is the previous day are found by default.
- On the Retroactive Data Generation Instance for Behavior Rule tab, click **Today** or **Yesterday** or select a date below **Retroactive Data Generation Date** to filter retroactive data generation instances. On the Retroactive Data Generation Instance for Tag tab, click Yesterday or 2 Days Ago or select a date below End Date to filter retroactive data generation instances.

View the operational logs of a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance whose operational logs you want to view. In the DAG in the right-side workspace, right-click the instance node and select **View Operational Log**.
2. On the **Operational Log** page, view the operational logs of the instance.

View the code of a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance whose code you want to view. In the DAG in the right-side workspace, right-click the instance node and select **View Node Script**.
2. On the **Node Script** page, view the code of the instance.

View the operation logs of a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance whose operation logs you want to view. In the DAG in the right-side workspace, right-click the instance node and select **View Operations Log**.
2. In the **Operations Log** pane, view the operation logs of the instance. You can click the icon in the **Details** column to view the operation details.

Terminate a retroactive data generation instance

Note You can terminate a retroactive data generation instance only when the instance is in the **Wait Submission** or **Running** state.

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance that you want to terminate. In the DAG in the right-side workspace, right-click the instance node and select **Terminate**.
2. In **Terminate** message, click **OK**.

Rerun a retroactive data generation instance

Note
- You can rerun a retroactive data generation instance only when the instance is in the **Success**, **Not Running**, or **Failed** state.
- Use caution when you rerun a retroactive data generation instance, because the data generated by the instance previously will be overwritten after the instance is rerun.

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance that you want to rerun. In the DAG in the right-side workspace, right-click the instance node and select **Rerun and Resume Scheduling**.
2. In the **Rerun and Resume Scheduling** message, click **OK**.

Rerun the downstream nodes of a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance whose downstream nodes you want to rerun. In the DAG in the right-side workspace, right-click the instance node and select **Rerun Downstream Nodes**.
2. In the **Rerun Downstream Nodes** dialog box, select the nodes to be rerun and click **OK**.

Set the running status of a retroactive data generation instance to Success and schedule the instance

Note You can set the running status of a retroactive data generation instance to Success and schedule the instance only when the instance is in the **Failed** state.

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance for which you want to set the running status to Success for scheduling. In the DAG in the right-side workspace, right-click the instance node and select **Set to Succeeded and Resume Scheduling**.
2. In the **Set to Succeeded and Resume Scheduling** message, click **OK**.

Forcibly rerun a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance that you want to forcibly rerun. In the DAG in the right-side workspace, right-click the instance node and select **Force Rerun**.
2. In the **Force Rerun** message, click **OK**.

> **Note** Use caution when you rerun a retroactive data generation instance, because the data generated by the instance previously will be overwritten after the instance is rerun.

Pause a retroactive data generation instance

1. In the **Retroactive Data Generation Instances** section, click the retroactive data generation instance that you want to pause. In the DAG in the right-side workspace, right-click the instance node and select **Pause**.
2. In the **Pause** message, click **OK**.

9.13. Monitoring and alerting

9.13.1. Alert records

When a monitored task triggers an alert, Dataphin pushes an alert notification to the specified recipients by using the notification method you specify in your task monitoring configuration. An alert record appears on the Alerts in Task page. You can view historical alerts on this page.

Search for and view alert records

1. On the Dataphin homepage, click **R&D** in the top navigation bar. On the R&D page, click **Scheduling** in the top navigation bar. On the page that appears, click **Monitoring** in the left-side navigation pane and click **Alerts in Task** on the left-side navigation submenu. If the current project is in Dev-Prod mode, you can only perform monitoring and alerting operations in Prod mode.
2. In the search box at the top of the workspace, you can enter keywords to search for alert records. Click the drop-down arrow to show more filter conditions. You can filter the search results by selecting filter conditions such as the alert cause, notification method, and notification status.

   Valid options of **Notification Status** include **Success**, **Sending**, and **Failed**.

   - **Success** indicates that an alert notification has been sent by using the specified notification method, and a receipt is returned.
   - **Sending** indicates that an alert notification has been sent by using the specified notification method, but no receipt is returned.
   - **Failed** indicates that one of the following errors occurs:
     - An alert notification cannot be sent by using the specified notification method.

       The error may result from one of the following causes: The email address or webhook URL has not been specified. The number of emails or DingTalk messages that the tenant can receive on the current day has reached the upper limit. The number of emails or DingTalk messages that the tenant can send on the current day has reached the upper limit. The specified recipient cannot be found in the contact information of the tenant because the selected recipient has been deleted.
     - An alert notification has been sent by using the specified notification method, but an error is reported.
     - A system error occurred. You can view the cause of the failure to obtain the detailed error message.

3. In the lower part of the workspace, you can view the search results, including the task name and task ID, monitoring type, alert cause, notification method, notification status, recipient, notification time, and available actions of each alert. By default, the alert records are sorted in descending order by notification time. You can click the icon to adjust the order of the alert records.
   - You can click the icon to view the new alert records.
   - You can click the icon to view the details of an alert.
   - You can move the pointer over the icon next to **Failed** of an alert record to view the cause of the failure.

   > **Note** If you have not configured the contact information for the recipient or the contact information is incorrect, **Failed** appears in the Notification Status column. You can move the pointer over the icon next to the notification status to view the cause of the failure.

Additional instructions

The maximum number of alert notifications that can be sent by using each notification method is limited. For more information, see Configure task monitoring.

> **Note** During the public preview, the monitoring and alerting feature of Dataphin is temporarily unavailable. You cannot configure the contact information, that is, pushing alert notifications is not supported.

9.13.2. Manage task monitoring configurations

This topic describes how to search for and view existing task monitoring configurations, enable and disable task monitoring, and modify or delete task monitoring configurations.

Search for and view existing task monitoring configurations

1. On the Dataphin homepage, click **R&D** in the top navigation bar. On the R&D page, click **Scheduling** in the top navigation bar. On the page that appears, click **Monitoring** in the left-side navigation pane and click **Task Monitoring Configuration** on the left-side navigation submenu.
2. On the Task Monitoring Configuration page:
   - In the search box at the top of the workspace, you can enter keywords to search for task monitoring configurations. Click the drop-down arrow to show more filter conditions. You can filter the search results by selecting the alert cause, notification method, alert notification recipient, and creator.
   - In the lower part of the workspace, you can view the search results, including the task name and task ID, alert cause, notification method, alert notification recipient, and creator of each task monitoring configuration.

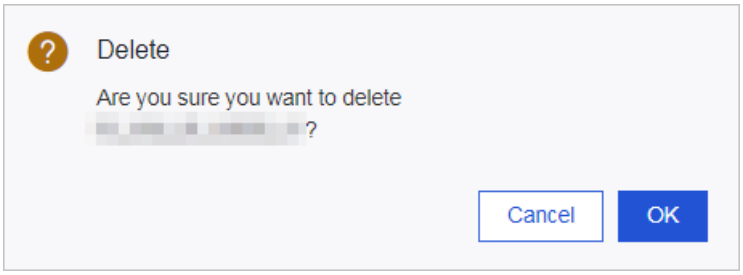
Enable and disable task monitoring

In the search result section, you can enable or disable monitoring for a specific task by turning on or off **Monitor Switch** for the target task monitoring configuration.

Modify or delete task monitoring configurations

In the search result section, click the **Change** icon in the **Actions** column for a task monitoring configuration. In the dialog box that appears, you can modify the alert cause, alert notification recipient, and notification method.

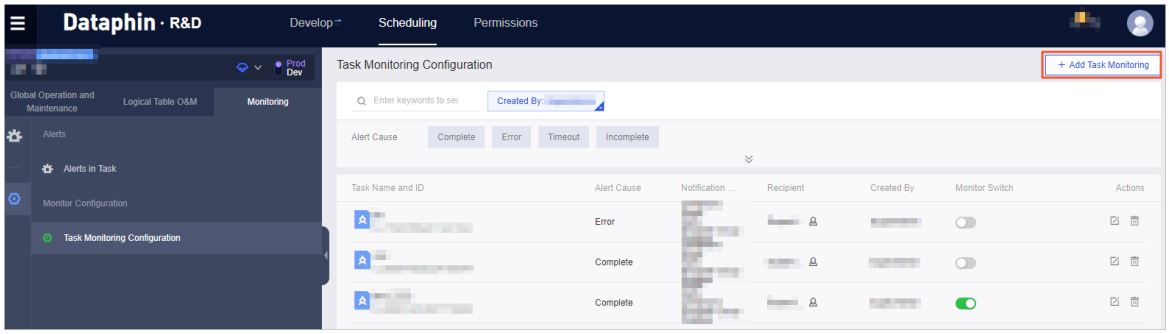
Click the **Delete** icon in the **Actions** column for a task monitoring configuration. A confirmation message appears, confirming whether you want to delete the task monitoring configuration.



9.13.3. Configure task monitoring

You can configure task monitoring to monitor the running status of tasks published to the production environment. This helps you know the task status in real time. This topic describes how to configure task monitoring.

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **R&D** in the top navigation bar. On the R&D page, click **Scheduling** in the top navigation bar. On the page that appears, click **Monitoring** in the left-side navigation pane and click **Task Monitoring Configuration** on the left-side navigation submenu.



> **Note** If the current project is in Dev-Prod mode, you must click the ▫ icon next to the project name in the left-side navigation pane to switch to the Prod mode. You can only perform monitoring and alerting operations in Prod mode.

3. On the Task Monitoring Configuration page, click **Add Task Monitoring** in the upper-right corner.
4. In the Add Task Monitoring dialog box that appears, set the parameters as prompted.
   - Select Task: the task to be monitored. You can select one or more tasks from the drop-down list for monitoring.

- Alert Cause: the trigger condition for the alert. You can select one of the following options: Error, Complete, Incomplete (until a certain time point), and Timeout (of specified minutes).
- Recipient: the user who receives the alert notification. You can specify the owner of the task or another user. One to three users can be specified at a time.
- Notification Method: the alert notification method. You can select Email, SMS, DingTalk Group Chatbot, or Cellphone.

  > Note
  > - A tenant can send a maximum of 100 messages by SMS, 2,000 messages by email, and 2,000 messages by DingTalk chatbot per day.
  > - A tenant can receive a maximum of 20 messages by SMS, 400 messages by email, and 400 messages by DingTalk chatbot per day.
  > - A tenant can send a maximum of 100 voice messages per day. Voice messages that are shorter than 1 minute is charged for 1 minute.

5. Click **OK**.

> Note During the public preview, the monitoring and alerting feature of Dataphin is temporarily unavailable. You cannot configure the contact information, that is, pushing alert notifications is not supported.

9.13.4. Alert rules for stream processing tasks

In Dataphin, you can configure alert rules to monitor the running status of stream processing tasks in the production environment. This topic describes how to create, modify, delete, enable, and disable alert rules for stream processing tasks.

Go to the Stream Processing Monitor Settings page

1. Log on to the Dataphin console.
2. Go to the **Monitoring** tab.
   - If the project that you accessed last time is in **Prod** or **Basic** mode, use one of the following methods to go to the **Monitoring** tab:
     - a. On the Dataphin homepage, click **R&D** in the top navigation bar.
       b. On the **Develop** page, click **Scheduling** in the top navigation bar.
       c. On the **Scheduling** page, click the **Monitoring** tab in the left-side navigation pane.
     - a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner.
       b. On the **Scheduling** page, click the **Monitoring** tab in the left-side navigation pane.
   - If the project that you accessed last time is in **Dev** mode, use one of the following methods to go to the **Monitoring** page:

- a. On the Dataphin homepage, click **R&D** in the top navigation bar.
  b. On the **Develop** page, click the icon next to the project name in the upper-left corner and select a project in **Prod** or **Basic** mode.
  c. On the **Develop** page, click **Scheduling** in the top navigation bar.
  d. On the **Scheduling** page, click the **Monitoring** tab in the left-side navigation pane.
- a. On the Dataphin homepage, click **Scheduling and Management** in the middle section, or click **Enter** next to **Scheduling** in the upper-right corner.
  b. On the **Scheduling** page, click the icon next to the project name in the upper-left corner and select a project in **Prod** or **Basic** mode.
  c. On the **Scheduling** page, click the **Monitoring** tab in the left-side navigation pane.

3. On the **Monitoring** tab, click the icon on the left-side navigation submenu. The **Stream Processing Monitor Settings** page appears.

   On the **Stream Processing Monitor Settings** page, you can filter alert rules for stream processing tasks by alert cause, notification method, recipient, and alert rule creator. You can also search for alert rules by entering a keyword of task names in the search box.In the lower part of the **Stream Processing Monitor Settings** page, you can view alert rules for stream processing tasks in detail, including the task name and ID, alert cause, notification method, recipient, alert rule creator, monitoring status, and actions that you can perform on each alert rule.

Parameter	Description
Task Name and ID	The name and ID of the stream processing task that is monitored based on the alert rule.
Alert Cause	The cause of alerts. Valid values: ○ **Excessively Long Delay** ○ **Exceeds Maximum TPS** ○ **Exceeds Specified Failure Frequency** ○ **Exceeds Specified Data Retention Period**
Notification Method	The method to notify the alert recipients. Valid values: ○ **Email** ○ **SMS** ○ **Cellphone** ○ **DingTalk Group Chatbot**
Recipient	The users to whom the alert notifications will be sent.
Created By	The user who created the alert rule.
Last Modified At	The time when the alert rule was last modified.

Parameter	Description
Monitor Switch	Indicates whether the alert rule is enabled for the task. ○ : The alert rule is enabled for the task. ○ : The alert rule is disabled for the task. You can click the switch to enable or disable the alert rule for the task. You can also click the icon next to the switch to delete the alert rule.
Actions	The actions that you can perform on the alert rule. Valid values: ○ You can click the icon to modify the alert rule. ○ You can click the icon to delete the alert rule.

Create an alert rule for a stream processing task

1. On the **Stream Processing Monitor Settings** page, click **Create Stream Processing Task Monitor** in the upper-right corner.
2. In the **Create Stream Processing Task Monitor** dialog box, set the parameters as required.

Section	Parameter	Description
Select Task	Task	The stream processing task to be monitored.
	Alert Causes	The alert cause of the alert rule. Click the icon next to **Alert Causes** to add an alert rule with another alert cause. You can configure the following alert rules for a stream processing task: ○ **Exceeds Specified Data Retention Period**: You must specify an upper limit for the queued time, in seconds. When the queued time exceeds this limit, an alert is triggered. ○ **Excessively Long Delay**: You must specify an upper limit for the processing delay, in seconds. When the processing delay exceeds this limit, an alert is triggered. ○ **Exceeds Maximum TPS**: You must specify a lower limit and an upper limit for the TPS. When the TPS is beyond the specified range, an alert is triggered. ○ **Exceeds Specified Failure Frequency**: You must specify an upper limit for the number of failures that occur in a minute. If the number of failures that occur in a minute exceeds this limit, an alert is triggered. You can click the icon next to an alert rule to delete the alert rule.

Section	Parameter	Description
Specify Alert Causes	Alerted Every	The time interval between alerts, in minutes. Valid values: 1 to 59.
	Time Range	The period during which the task is monitored. Valid values: ○ All Day ○ Specified
	Recipients	The users to whom the alert notifications will be sent. Valid values: ○ Owner: the owner of the stream processing task. ○ Custom: the specified users to whom the alert notifications will be sent. You can specify a maximum of five alert recipients for a stream processing task. ○ Shift Schedule: the shift schedule of users to whom the alert notifications will be sent. You can specify a maximum of one shift schedule.
	Notification Method	The method to notify the alert recipients. Valid values: ○ Email ○ SMS ○ Cellphone ○ DingTalk

Note

- For an Alibaba Cloud account, the system can send a maximum of 100 messages by SMS, 2,000 messages by email, and 2,000 messages by DingTalk chatbot per day.
- An Alibaba Cloud account can receive a maximum of 20 messages by SMS, 400 messages by email, and 400 messages by DingTalk chatbot per day.
- For an Alibaba Cloud account, the system can send a maximum of 100 voice messages per day. A voice message that is shorter than 1 minute is billed as 1 minute.

3. Click **OK**.

Modify an alert rule for a stream processing task

1. On the **Stream Processing Monitor Settings** page, find the alert rule that you want to modify and click the icon in the **Actions** column.
2. In the **Edit Stream Processing Task Monitor** dialog box, modify the parameters as required. For more information, see Create an alert rule for a stream processing task.
3. Click **OK**.

Delete alert rules for a stream processing task

1. On the **Stream Processing Monitor Settings** page, find the alert rules that you want to delete and click the icon in the **Actions** column.
2. In the **Tip** message, click **OK**.

Enable or disable an alert rule for a stream processing task

On the **Stream Processing Monitor Settings** page, find the alert rule that you want to enable or disable and click the or icon in the **Monitor Switch** column.

Delete an alert rule for a stream processing task

1. On the **Stream Processing Monitor Settings** page, find the alert rule that you want to delete and click the icon in the **Monitor Switch** column.
2. In the **Tip** message, click **OK**.

9.14. Data assets

9.14.1. Overview

This topic describes the Data Assets module of Dataphin and the three modes of viewing data assets, that is, the Global, Flow, and Structure modes.

Data asset overview

After development works such as data collection, integration, and processing are completed, you can manage data assets in a systematic way.

Based on the standards and methodology of data asset management, the Data Assets module of Dataphin allows you to take inventory of and evaluate the data assets in your enterprise, including:

- Automatically extracts and analyzes metadata, and creates a data asset dashboard. This helps enterprise managers discover and understand the value of data assets.
- Provides an end-to-end inventory check and analysis of computing, storage, security, and applications during data production. This helps you detect problems, propose and implement governance optimization schemes, reduce costs, and improve efficiency for data.
- Provides the Data Assets page for you to view the data modeling results and table details.

You can view data assets in **Global**, **Flow**, and **Structure** modes.

- The Global mode displays business units that contain a large amount of data in the form of planets and their respective data sizes.
- The Flow mode displays the entire process of data ingestion, integration, and output. This reveals the underlying potential capabilities of a Data Mid-End.
- The Structure mode displays components in different shapes to represent business entities and uses lines of different styles to represent relationships between these entities. This mode clearly shows the structure of data in a business unit.

Global mode

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the Data Assets page, click **Overview** in the top navigation bar.
4. On the **Overview** page, click **Global** in the upper-right corner.
5. The **Global** tab displays the total number of tasks, tables, and projects in Dataphin in the lower-right corner. It also displays the rankings of business units that contain a large amount of data, including the business unit name, data size, and proportion to the total data size.You can also move the pointer over a planet to view the statistics about the business unit, including the amount of used computing resources and storage size.

> **Note** The **Overview** page only displays information about the metadata of the production environment and the application data that has been processed.

Flow mode

On the **Overview** page, click **Flow** in the upper-right corner.

The **Flow** tab displays the data ingestion progress, the total number of tables, and data applications such as data query in a visual view.

- Move the pointer over **Data Ingestion Progress** to view the number of data sources.
- Move the pointer over **Total Tables** to view the statistics about tables from the perspectives of data standardization and data models.

Structure mode

On the **Overview** page, click **Structure** in the upper-right corner. On the **Structure** tab, click a business unit in the upper-right corner to view all the dimensions, business processes, and their relationships under this business unit.The **Structure** tab displays the following information in the upper-left corner:

- **Dimensions**: the types of dimensions. Dimensions are classified into common dimensions and other dimensions. Other dimensions include common dimensions by hierarchy, enumeration dimensions, and virtual dimensions.
- **Relation**: the relationships between dimensions.
  - indicates a parent-child relationship.
  -  indicates an association relationship.
- Search box: You can enter a dimension or business process name in the **Search** field or click the  icon next to **Search**, and select a dimension or business process from the drop-down list. The dimensions and business processes related to the selected dimension or business process appear.

You can use one of the following methods to view relationships between dimensions and between dimensions and business processes:

- Click a dimension. The dimensions and business processes related to this dimension are highlighted.
- Click a business process. The dimensions related to this business process are highlighted.
- Use the search box in the upper-left corner of the **Structure** tab to search for the desired

dimension or business process and view its related dimensions and business processes.

9.14.2. Map

9.14.2.1. View the asset map

An asset map summarizes the relationships between dimensions and business processes in a data domain of a business unit to show the structure of your enterprise data. In addition, the asset map allows you to efficiently and accurately search for and explore data based on features such as data search, access recording, and bookmarking.

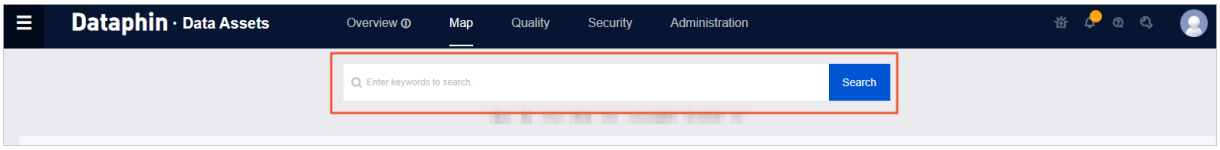
Map page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar. The **Map** page appears by default.

   The Map page consists of the following four sections:

   - Search bar
   - Business Units section
   - My Data Assets section
   - Recommended section

Search bar



- You can enter a keyword in the search box to search for tables whose names contain the keyword in the development or production environment.
- By default, when you enter a keyword in the search box, 10 related tables and multiple related keywords that match the keyword appear in the drop-down list. You can view the names, display names, and descriptions of the related tables in the drop-down list.
- If you select a related table, the table details page appears. If you select a related keyword, the search results are updated based on the selected keyword.

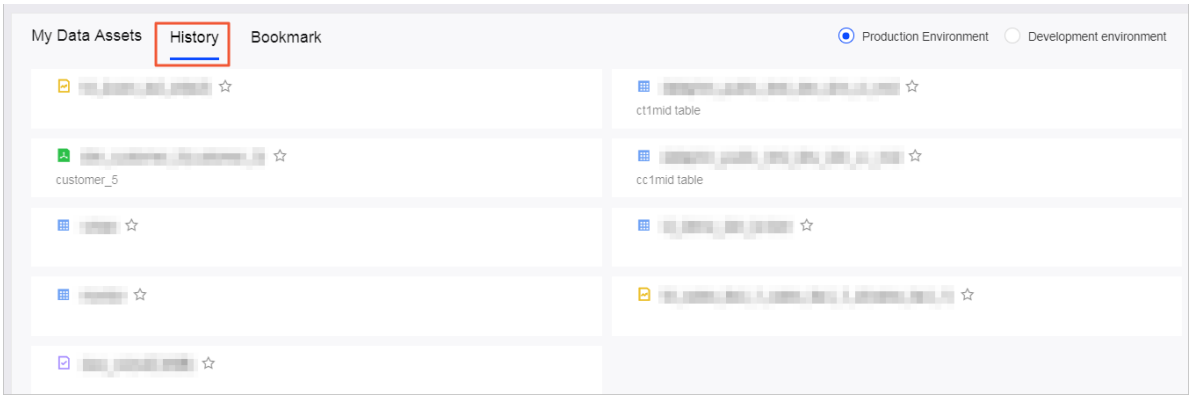
Business Units section

The **Business Units** section is below the search bar. In this section, the division of information echoes the asset overview. This section only displays the data of the production environment, including the business units, data domains, dimensions, and business processes.

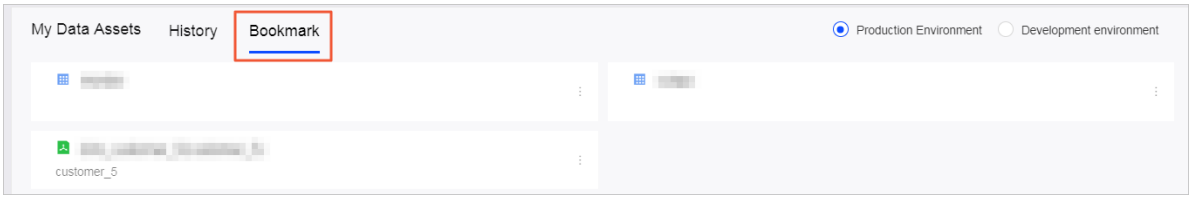
My Data Assets section

This section displays the data assets that you have accessed or bookmarked. You can switch between the development environment and production environment to view the access history and bookmarks. By default, the My Data Assets section displays the access history and bookmarks of the production environment.

- The **History** tab displays the 10 data assets that you recently accessed.

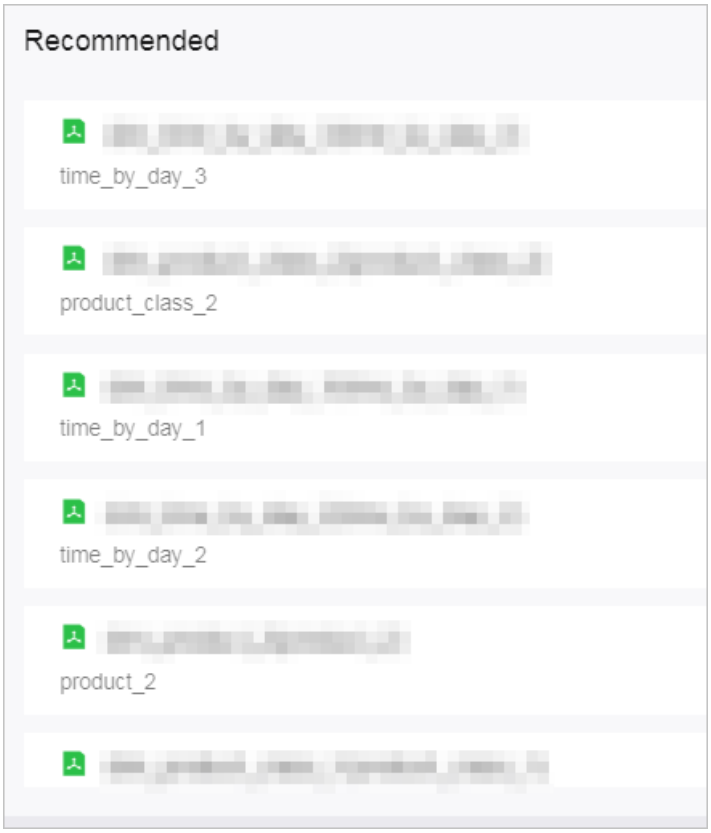


- The **Bookmark** tab displays the data assets that you bookmarked on the asset details page. The number of data assets that can be bookmarked is not limited.



Recommended section

This section displays the most popular tables, including their names, display names, and descriptions, based on the popularity of global user access.



Metadata

No search results will be returned if you query a table that is not created in Dataphin or a newly created table whose metadata has not been synchronized because of the latency of Dataphin in obtaining metadata. In this case, you can manually synchronize the metadata of the table to Dataphin.

1. On the **Map** page, click **Search** in the search bar.
2. On the page that appears, click the icon in the upper-right corner. The **Refresh Metadata** dialog box appears.
3. Set Project Name and Table Name and click **OK**.
   - **Project Name**: the name of the project to which the table belongs. You can view all projects of the current tenant in the drop-down list. Select one from the list.
   - **Table Name**: the name of the table whose metadata needs to be synchronized to Dataphin. You must enter an exact match to the table name.

     > **Note** Make sure that the table name is correct before you click **OK**.

9.14.2.2. View table details

This topic describes how to search for physical tables, logical tables, and stream metatables, and view table details.

Search for tables

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar. The **Map** page appears by default.
3. Enter a keyword in the search box and press Enter or click **Search**. Alternatively, select a dimension or business process in the Business Units section. The Search Results page appears.

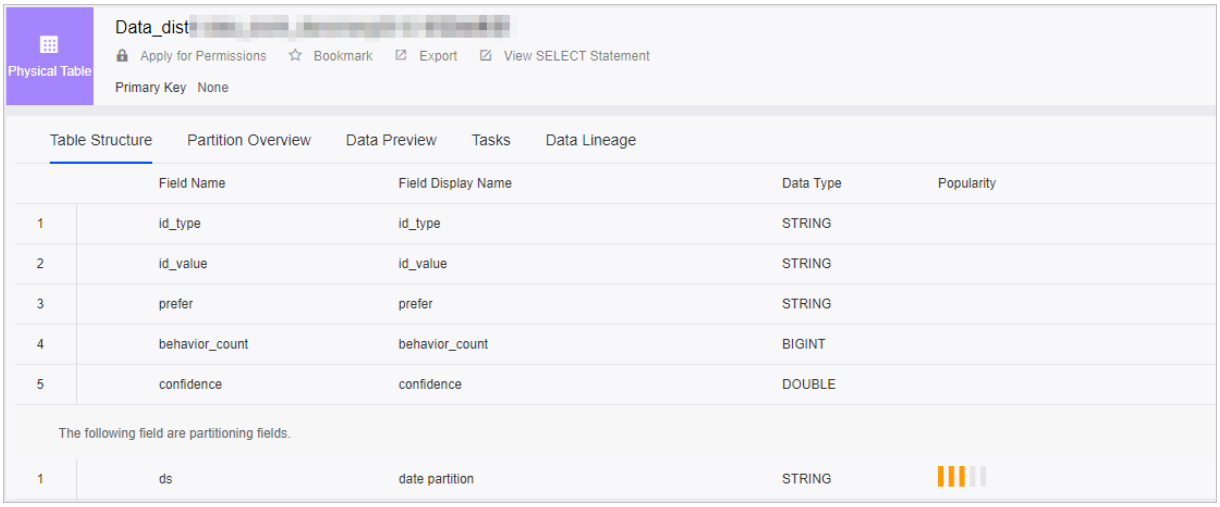
   The Search Results page provides the following information and features:

   - The left-side pane displays filter conditions and the right-side pane displays the tables that match the selected filter conditions. Filter conditions fall into the following two categories:
     - **Business Properties**: Filter conditions in this category are further classified by project, business unit, data domain, dimension, and business process.
     - **Data Properties**: Filter conditions in this category are further classified by environment, storage type, and table type, such as logical dimension table, logical fact table, and logical aggregate table.
   - You can select one or more filter conditions in the left-side pane to filter tables and update the search results in the right-side pane. By default, Dataphin displays only one row of filter conditions for each type. You can click the downwards arrow to view more filter conditions. You can click the name of a table in the right-side pane to go to the details page of the table.

View the details about a physical table

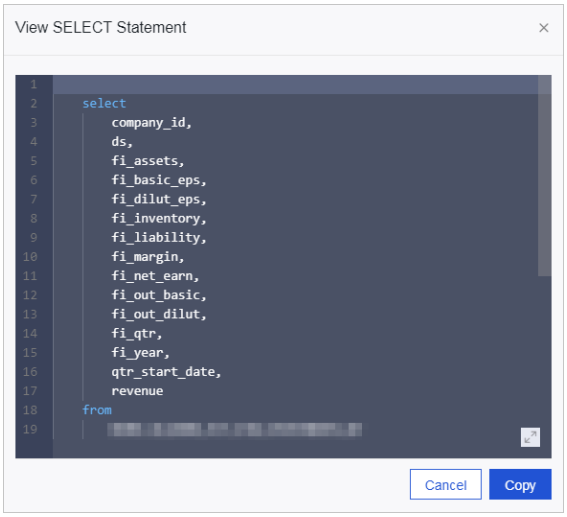
> **Note** The table details page varies with the table type.

On the Search Results page, click the name of a physical table. The details page of the table appears, as shown in the following figure.



- The upper-left part of the details page provides the following information and features:
  - Table type: The type of a table can be Logical Dimension Table, Logical Fact Table, Logical Aggregate Table, Physical Table, or Stream Metatable. In this example, the table type is **Physical Table**.
  - Table name: The table name is in the format of Business unit name. Table name. A maximum of 70 characters can be displayed for the name. The excess part is displayed as an ellipsis (...).
  - **Apply for Permissions**: Click Apply for Permissions. On the Apply for Physical Table Permissions tab, set the parameters to apply for required permissions.
  - **Bookmark** or Remove Bookmark: The action varies depending on whether the table is bookmarked.
    - The ☆ icon indicates that you have not bookmarked the table. You can click the ☆ icon to bookmark the table.
    - The ★ icon indicates that you have bookmarked the table. You can click the ★ icon to remove the table from your bookmarks.
  - **Export**: You can click **Export** to export the fields in the table to a CSV file.

- **View SELECT Statement**: Click **View SELECT Statement**. In the **View SELECT Statement** dialog box, you can view the SQL statement for querying the table data.



- In the right-side pane of the details page, you can view the basic information, access information, physical information, and update information about the table.

The lower-left part of the details page provides the following tabs:

- **Table Structure**: This tab displays the name, display name, data type, and popularity of the fields in the physical table.
- **Partition Overview**: This tab displays the partition information about the physical table.
- **Data Preview**: This tab displays the table content, including the name and records of each metric.

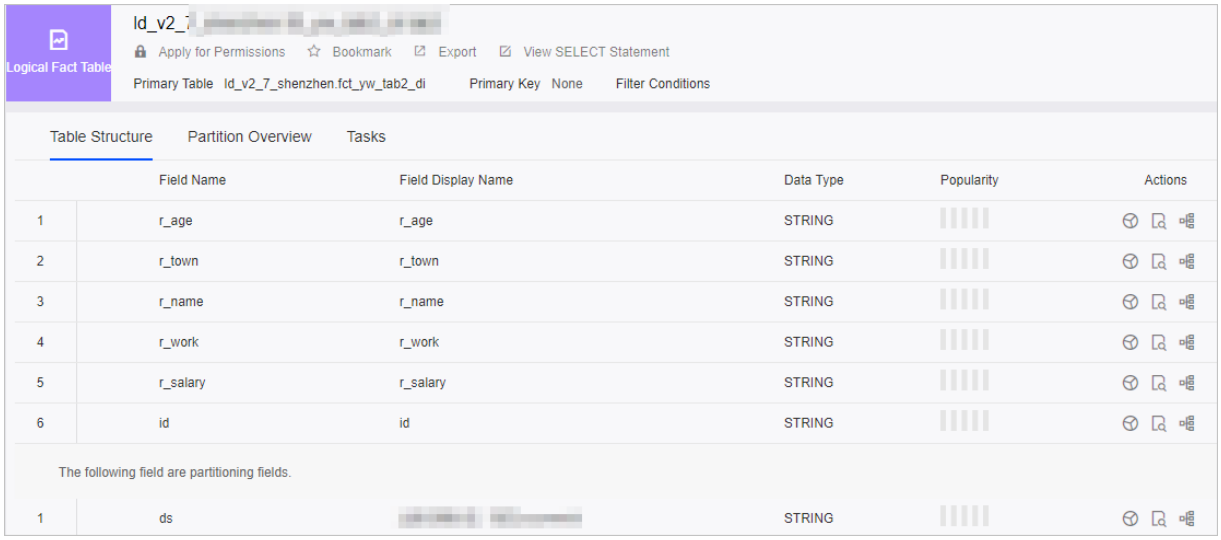
  > **Note** If you are not authorized to view the records, each record is de-identified with multiple asterisks (*****).

- **Tasks**: This tab displays the scheduling information about the tasks that are related to the physical table, including the ID, execution time in seconds, start time, and end time of each task.
  - You can click the icon in the **Task ID** column of a task to go to the **Recurring Task Instances** page. On this page, you can view the instances that have been generated for the task.
  - You can click the icon in the **Task ID** column of a task to go to the **Node Script** page. On this page, you can view the code of the task.
- **Data Lineage**: This tab displays the dependencies between tables.

View the details about a logical table

> **Note** This section describes how to view the details about a logical fact table. Follow the same procedure for other types of logical tables.

On the Search Results page, click the name of a logical fact table. The details page of the table appears, as shown in the following figure.

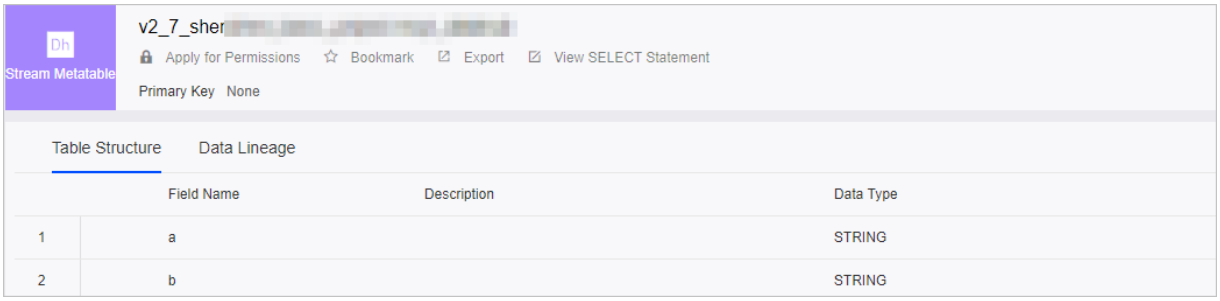


The details page consists of the upper-left part, lower-left part, and right-side pane.

- In the upper-left part, you can view the type, name, primary table, primary key, and filter conditions of the logical fact table. You can also perform the following actions on the logical fact table: **Apply for Permissions**, **Bookmark** or Remove Bookmark, **Export**, and **View SELECT Statement**.
- In the right-side pane, you can view the basic information, access information, physical information, and update information about the logical fact table.
- The **Table Structure** tab in the lower-left part displays the name, display name, data type, and popularity of the fields in the logical fact table. On this tab, you can perform the following operations:
  - Click the icon in the **Actions** column of a field. In the dialog box that appears, you can view the partitions, records, creation time, and update time of the field.
  - Click the icon in the **Actions** column of a field. In the dialog box that appears, you can view the numerical distribution and numerical interval distribution of the field, as well as the data records.
  - Click the icon in the **Actions** column of a field. In the dialog box that appears, you can view the data lineage of the field.
- The **Partition Overview** tab in the lower-left part displays the distribution of partitions for each field in the logical fact table. You can filter data by field or partition.
- The **Tasks** tab in the lower-left part displays the scheduling information about the tasks that are related to the logical fact table, including the ID, execution time in seconds, start time, and end time of each task. On this tab, you can perform the following operations:
  - Click the icon in the **Task ID** column of a task to go to the **Recurring Task Instances** page. On this page, you can view the instances that have been generated for the task.
  - Click the icon in the **Task ID** column of a task to go to the **Node Script** page. On this page, you can view the code of the task.

View the details about a stream metatable

On the Search Results page, click the name of a stream metatable. The details page of the table appears, as shown in the following figure.



The details page consists of the upper-left part, lower-left part, and right-side pane.

- By default, the upper-left part displays the type, name, and primary key of the stream metatable.
- In the right-side pane, you can view the basic information, access information, physical information, and update information about the stream metatable.
- The **Data Lineage** tab in the lower-left part displays the data lineage of the stream metatable.
- The **Table Structure** tab in the lower-left part displays the fields in the stream metatable.

  If you have the query permission on the table, you can view the first 10 fields in the table. If you do not have the query permission on the table, click **Apply for Permissions** to apply for permissions on the table. For more information, see Manage permissions on stream metatables.

9.14.3. Security

9.14.3.1. My permissions

9.14.3.1.1. Manage permissions on logical tables

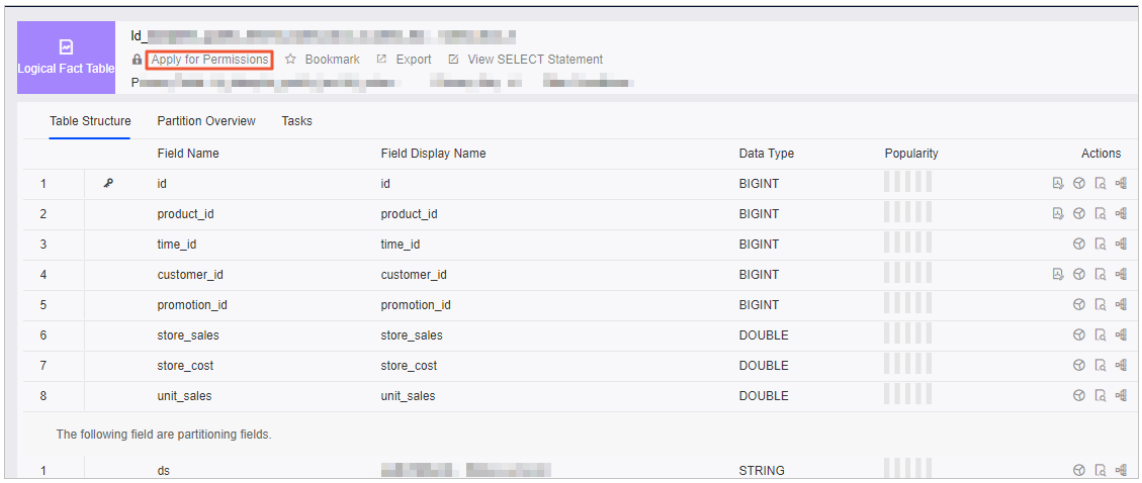
This topic describes how to apply for, query, renew, and remove permissions on logical tables.

Apply for permissions on a logical table

1. Log on to the Dataphin console.
2. Go to the **Logical Table Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Logical Table Permissions** under My Permissions in the left-side navigation pane.
3. On the **Logical Table Permissions** tab, go to the **Apply for Logical Table Permissions** tab by using one of the following methods:
   - Click **Apply for Logical Table Permissions** in the upper-right corner.
   - In the left-side navigation pane, move the pointer over **Logical Table Permissions** and

click the icon.

- In the left-side navigation pane, click the icon next to My Permissions and select **Logical Table Permissions**.
-



a. In the top navigation bar, click **Map**.

b. On the **Map** page, enter a keyword in the search box to search for tables whose names contain the keyword.

c. Find the logical table on which you want to apply for permissions and click the table name. On the details page that appears, click **Apply for Permissions** in the upper-left part.

4. On the **Apply for Logical Table Permissions** tab, perform the following operations in the configuration wizard:

i. In the **Select Data Objects** step, set the **Environment**, **Business Unit**, and **Logical Table** parameters.

ii. Click **Next**.

iii. In the **Apply for Permissions** step, set the parameters as required.

Parameter	Description
Authorized Fields	The fields on which you want to apply for permissions. Select fields by using one of the following methods: ■ Enter a keyword in the search box to search for the desired fields and select the fields. ■ Select fields from the field list of the logical table.
Permission Type	The permission that you want to apply for. By default, the value **Query** is selected and you cannot modify the setting.
Account Type	The accounts to be granted the permission that you are applying for. You can grant the permission to either or both of the personal account and project production account. ■ The personal account is used for data modeling and development in the development environment and publishing developed objects or tasks to the production environment. If you select **Personal Account**, you must also set the **Expiration Date** parameter. ■ The project production account is used to manage and run tasks that are submitted to the production environment. If you select **Project Production Account**, you must also set the **Project** parameter.
Reason for Application	The reason why you apply for the permission.

5. Click **Submit**.

Query the permissions on a logical table

1. On the **Logical Table Permissions** tab, find the logical table on which you want to query the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, view the fields on which you have applied for permissions and the specific permissions.

Renew the permissions on a logical table

1. On the **Logical Table Permissions** tab, find the logical table on which you want to renew the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, select the fields on which you want to renew permissions.
   - Enter a keyword in the search box to search for the desired fields and select the fields.
   - Select fields from the field list of the logical table.
3. In the lower part of the page, click **Extend Validity**.
4. In the **Apply for Permissions** step, set the parameters as required. For more information, see Apply for permissions on a logical table.
5. Click **Submit**.

Remove the permissions on a logical table

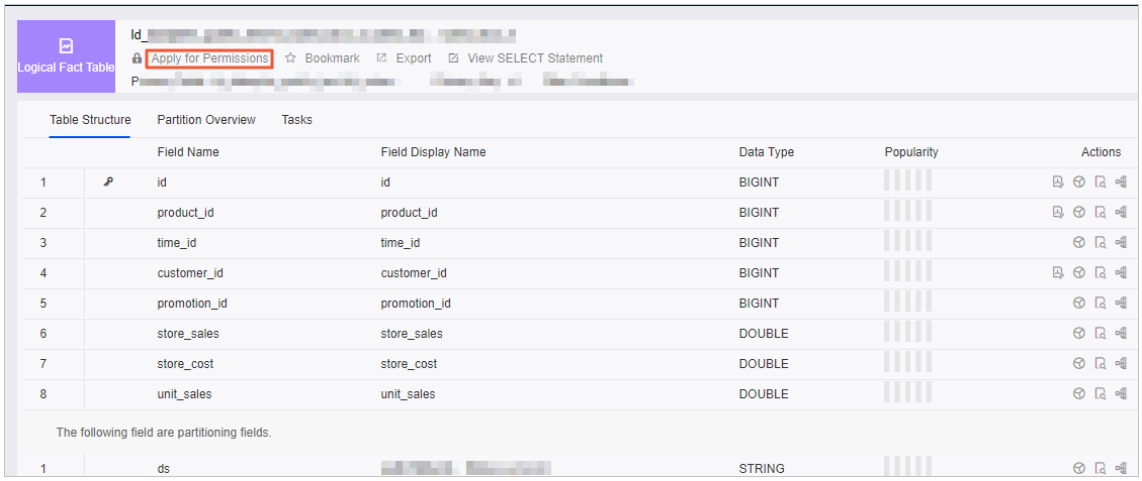
1. On the **Logical Table Permissions** tab, find the logical table on which you want to remove the permissions and click the icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.2. Manage permissions on physical tables

This topic describes how to apply for, query, renew, and remove permissions on physical tables.

Apply for permissions on a physical table

1. Log on to the Dataphin console.
2. Go to the **Physical Table Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Physical Table Permissions** under My Permissions in the left-side navigation pane.
3. On the **Physical Table Permissions** tab, go to the **Apply for Physical Table Permissions** tab by using one of the following methods:
   - Click **Apply for Physical Table Permissions** in the upper-right corner.
   - In the left-side navigation pane, move the pointer over Physical Table Permissions and click the icon.
   - In the left-side navigation pane, click the Add icon next to My Permissions and select **Physical Table Permissions**.
   -



   a. In the top navigation bar, click **Map**.
   b. On the **Map** page, enter a keyword in the search box to search for tables whose names contain the keyword.
   c. Find the physical table on which you want to apply for permissions and click the table name. On the details page that appears, click **Apply for Permissions** in the upper-left part.

4. On the **Apply for Physical Table Permissions** tab, perform the following operations in the configuration wizard:
   i. In the **Select Data Objects** step, set the **Environment**, **Project**, and **Physical Tables** parameters.
   ii. Click **Next**.
   iii. In the **Apply for Permissions** step, set the parameters as required.

Parameter	Description
Authorized Fields	The fields on which you want to apply for permissions. Select fields by using one of the following methods: ■ Enter a keyword in the search box to search for the desired fields and select the fields. ■ Select fields from the field list of the physical table.
Permission Type	The permissions that you want to apply for. Valid values: ■ **Query** ■ **Write** ■ **Update** ■ **Delete**
Account Type	The accounts to be granted the permissions that you are applying for. You can grant the permissions to either or both of the personal account and project production account. ■ The personal account is used for data modeling and development in the development environment and publishing developed objects or tasks to the production environment. ■ The project production account is used to manage and run tasks that are submitted to the production environment.
Reason for Application	The reason why you apply for the permissions.

5. Click **Submit**.

Query the permissions on a physical table

1. On the **Physical Table Permissions** tab, find the physical table on which you want to query the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, view the fields on which you have applied for permissions and the specific permissions.

Renew the permissions on a physical table

1. On the **Physical Table Permissions** tab, find the physical table on which you want to renew the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, select the fields on which you want to renew permissions.
   - Enter a keyword in the search box to search for the desired fields and select the fields.

- Select fields from the field list of the physical table.

3. In the lower part of the page, click **Extend Validity**.
4. In the **Apply for Permissions** step, set the parameters as required. For more information, see Apply for permissions on a physical table.
5. Click **Submit**.

Remove the permissions on a physical table

1. On the **Physical Table Permissions** tab, find the physical table on which you want to remove the permissions and click the icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.3. Manage permissions on stream metatables

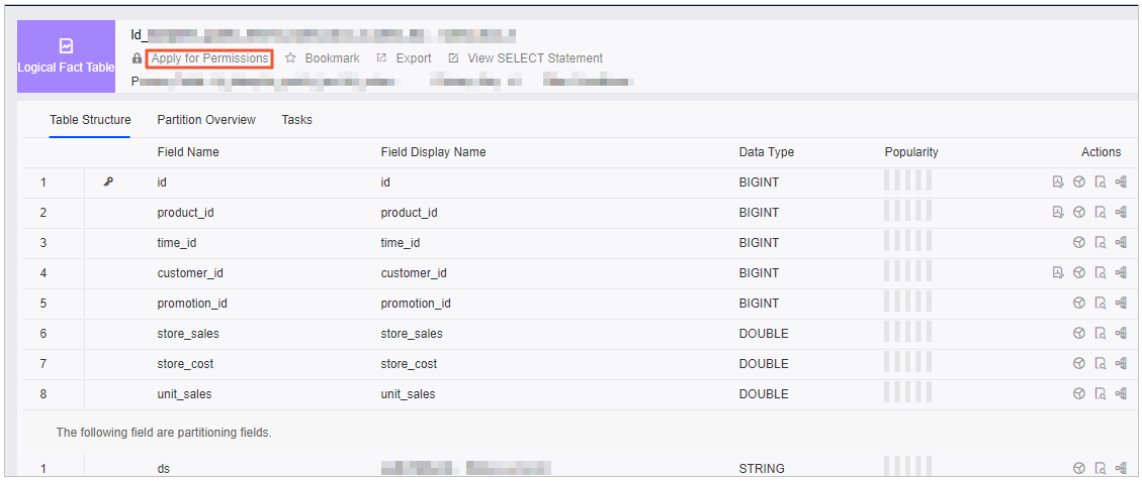
This topic describes how to apply for, query, renew, and remove permissions on stream metatables.

Context

A stream processing task in a project can reference a stream metatable in another project. If a stream processing task that you create needs to reference a stream metatable created by another user or in another project, you must apply for the query, write, update, or delete permission on the stream metatable.

Apply for permissions on a stream metatable

1. Log on to the Dataphin console.
2. Go to the **Stream Metatable Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Stream Metatable Permissions** under My Permissions in the left-side navigation pane.
3. On the **Stream Metatable Permissions** tab, go to the **Apply for Stream Metatable Permissions** tab by using one of the following methods:
   - Click **Apply for Stream Metatable Permissions** in the upper-right corner.
   - In the left-side navigation pane, move the pointer over Stream Metatable Permissions and click the icon.
   - In the left-side navigation pane, click the icon next to My Permissions and select **Stream Metadata Permissions**.
   - In the top navigation bar, click **Map**. On the **Map** page, enter a keyword in the search box to search for tables whose names contain the keyword. Find the stream metatable on which you want to apply for permissions and click the table name. On the details page that appears, click **Apply for Permissions** in the upper-left part.



4. In the **Select Data Objects** step, set the **Environment**, **Project**, and **Stream Metatables** parameters.
5. Click **Next**.
6. In the **Apply for Permissions** step, set the parameters as required.

Parameter	Description
Authorized Fields	The fields on which you want to apply for permissions. Select fields by using one of the following methods: ○ Enter a keyword in the search box to search for the desired fields and select the fields. ○ Select fields from the field list of the metatable.
Permission Type	The permissions that you want to apply for. Valid values: ○ **Query** ○ **Write** ○ **Update** ○ **Delete**
Account Type	The accounts to be granted the permissions that you are applying for. You can grant the permissions to either or both of the personal account and project production account. ○ The personal account is used for data modeling and development in the development environment and publishing developed objects or tasks to the production environment. ○ The project production account is used to manage and run tasks that are submitted to the production environment.
Reason for Application	The reason why you apply for the permissions.

7. Click **Submit**.

Query the permissions on a stream metatable

1. On the **Stream Metatable Permissions** tab, find the stream metatable on which you want to query the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, view the fields on which you have applied for permissions and the specific permissions.

Renew the permissions on a stream metatable

1. On the **Stream Metatable Permissions** tab, find the stream metatable on which you want to renew the permissions and click the table name in the **Data Object** column.
2. On the details page that appears, select the fields on which you want to renew permissions.
   - Enter a keyword in the search box to search for the desired fields and select the fields.
   - Select fields from the field list of the metatable.
3. In the lower part of the page, click **Extend Validity**.
4. In the **Apply for Permissions** step, set the parameters as required. For more information, see Apply for permissions on a stream metatable.
5. Click **Submit**.

Remove the permissions on a stream metatable

1. On the **Stream Metatable Permissions** tab, find the stream metatable on which you want to remove the permissions and click the  icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.4. Manage permissions on functions

This topic describes how to apply for and remove permissions on functions.

Context

When you use Dataphin for data warehouse modeling, you can use functions to improve the task development efficiency. You can manage the query permission on functions on the Function Permissions tab of Dataphin. If a task that you create needs to reference a function created by another user or in another project, you must apply for the query permission on the function.

Apply for the query permission on a function

1. Log on to the Dataphin console.
2. Go to the **Function Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Function Permissions** under My Permissions in the left-side navigation pane.
3. On the **Function Permissions** tab, go to the **Apply for Function Permissions** tab by using one of the following methods:
   - Click **Apply for Function Permissions** in the upper-right corner.
   - In the left-side navigation pane, click the  icon next to **Function Permissions**.

- In the left-side navigation pane, click the icon next to My Permissions and select Function Permissions.

4. On the **Apply for Function Permissions** tab, perform the following operations in the configuration wizard:
   i. In the **Select Data Objects** step, set the **Environment**, **Project**, and **UDF** parameters.
   ii. Click **Next**.
   iii. In the **Apply for Permissions** step, set the parameters as required.

Parameter	Description
Permission Type	The permission that you want to apply for. By default, the value **Query** is selected and you cannot modify the setting.
Account Type	The accounts to be granted the permission that you are applying for. You can grant the permission to either or both of the personal account and project production account. ■ The personal account is used for data modeling and development in the development environment and publishing developed objects or tasks to the production environment. If you select this option, you must also set the **Expiration Date** parameter. ■ The project production account is used to manage and run tasks submitted to the production environment. If you select this option, you must also set the **Project** parameter.
Reason for Application	The reason why you apply for the permission.

5. Click **Submit**.

Remove the query permission on a function

1. On the **Function Permissions** tab, find the function on which you want to remove the query permission and click the icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.5. Manage permissions on data sources

This topic describes how to apply for and remove permissions on data sources.

Context

After you obtain the read and write permissions on a data source, you have the following permissions:

- Permissions to read and write physical tables in the data source but not existing fields in the tables. For more information about how to apply for the read and write permissions on existing fields in the physical tables, see Manage permissions on physical tables.
- Permission to read logical tables in the data source but not existing fields in the tables. For more information about how to apply for the read permission on existing fields in the logical tables, see Manage permissions on logical tables.

Apply for permissions on a data source

1. Log on to the Dataphin console.
2. Go to the **Data Source Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Data Source Permissions** under **My Permissions** in the left-side navigation pane.
3. On the **Data Source Permissions** tab, go to the **Apply for Data Source Permissions** tab by using one of the following methods:
   - Click **Apply for Data Source Permissions** in the upper-right corner.
   - In the left-side navigation pane, click the icon next to **Data Source Permissions**.
   - In the left-side navigation pane, click the icon next to My Permissions and select **Data Source Permissions**.
4. On the **Apply for Data Source Permissions** tab, set the parameters as required.
   i. In the **Select Data Objects** step, set the **Environment**, **Project**, and **UDF** parameters.
   ii. Click **Next**.
   iii. In the **Apply for Permissions** step, set the parameters as required.

Parameter	Description
Data Source	The data source on which you want to apply for permissions.
Permission Type	The permissions that you want to apply for. Valid values: **Read** and **Write**.
Account Type	The accounts to be granted the permissions that you are applying for. You can grant the permissions to either or both of the personal account and project production account. ■ The project production account is used to periodically schedule production tasks in the project. Dataphin uses this type of account to isolate users who have a personal account from the production environment. If you select this option, you must also set the **Project** parameter. ■ The personal account is used to perform operations related to data standardization, data modeling, and coding in the development environment. If you select this option, you must also set the **Expiration Date** parameter.
Reason for Application	The reason why you apply for the permissions.

5. Click **OK**.

Remove the permissions on a data source

1. On the **Data Source Permissions** tab, find the data source on which you want to remove the

permissions and click the icon in the **Actions** column.

2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.6. Manage permissions on API operations

You can obtain permissions on API operations only after you activate the Services module. This topic describes how to apply for, query, and remove permissions on API operations.

Prerequisites

The Services module is activated.

Apply for permissions on an API operation

1. Log on to the Dataphin console.
2. Go to the **API Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **API Permissions** under My Permissions in the left-side navigation pane.
3. Go to the **API Permission Request** page in Application Center of the Services module by using one of the following methods:
   - On the **API Permissions** tab, click **Apply for API Permissions** in the upper-right corner.
   - In the left-side navigation pane, click the icon next to **API Permissions**.
   - In the left-side navigation pane, click the icon next to My Permissions and select **API Permissions**.
4. On the **API Permission Request** page, set the parameters in the configuration wizard. For more information, see Apply for permissions on an API operation.

Query permissions on an API operation

1. On the **API Permissions** tab, find the API operation on which you want to query permissions and click the name in the **Data Object** column.
2. On the details page, view the permission information about the API operation.

Remove the permissions on an API operation

1. On the **API Permissions** tab, find the API operation on which you want to remove the permissions and click the icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.1.7. Manage functionality permissions

This topic describes how to apply for and remove functionality permissions.

Apply for a functionality permission

1. Log on to the Dataphin console.
2. Go to the **Functionality Permissions** tab.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Security** in the top navigation bar.
   iii. On the **Permissions** page, click **Functionality Permissions** under **My Permissions** in the left-side navigation pane.
3. The **Functionality Permissions** tab does not provide the entry for you to apply for functionality permissions. You can apply for the corresponding functionality permissionif you are prompted to apply for the permission when you attempt to open a page, for example, the Overview page of the Data Assets module.
4. Click **Apply for Permission**.
5. In the **Apply for Functionality Permission** dialog box, set the **Expiration Date** and **Reason for Application** parameters.
6. Click **OK**.

Remove a functionality permission

1. On the **Functionality Permissions** tab, find the functionality on which you want to remove the permission and click the icon in the **Actions** column.
2. In the **Remove Permissions** message, click **OK**.

9.14.3.2. Managed permissions

9.14.3.2.1. Grant and revoke permissions on business units

This topic describes how to grant and revoke permissions on business units.

Grant permissions on a business unit to specific accounts

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Security** in the top navigation bar.
4. On the **Permissions** page, click **Business Unit Permissions** under Managed Permissions in the left-side navigation pane.
5. On the **Business Unit Permissions** tab, find the business unit on which you want to grant permissions to specific accounts and click the icon in the **Actions** column.
6. In the dialog box that appears, set the parameters as required.

Parameter	Description

Parameter	Description
Grant To	The accounts to which you want to grant permissions on the business unit. You can grant permissions on the business unit to either or both of the personal account and project production account. If you select an account from the **Personal Account** drop-down list, you must also set the **Expiration Date** parameter.
Logical Tables	The names of the logical tables on which you want to grant permissions to the specified accounts. You can select logical tables only in the current business unit.
Permission Type	The permissions to be granted. By default, the value **Query** is selected and you cannot modify the setting.
Reason for Grant of Permissions	The reason why you grant the permissions.

7. Click **OK**.

Revoke permissions on a business unit from a specific account

1. On the **Business Unit Permissions** tab, find the business unit on which you want to revoke permissions from a specific account and click the icon in the **Actions** column.
2. In the dialog box that appears, set the parameters as required.

Parameter	Description
Revoke From	The account from which you want to revoke permissions on the business unit.
Logical Tables	The names of the logical tables on which you want to revoke permissions from the specified account. You can select logical tables only in the current business unit.
Permission Type	The permissions that you want to revoke. By default, the value **Query** is selected and you cannot modify the setting.
Reason for Revocation of Permissions	The reason why you revoke the permissions.

3. Click **OK**.

9.14.3.2.2. Grant and revoke permissions on projects

This topic describes how to grant and revoke permissions on projects.

Grant permissions on a project to specific accounts

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.

3. On the **Data Assets** page, click **Security** in the top navigation bar.
4. On the **Permissions** page, click **Project Permissions** under Managed Permissions in the left-side navigation pane.
5. On the **Project Permissions** tab, find the project on which you want to grant permissions to specific accounts and click the icon in the **Actions** column.
6. In the dialog box that appears, set the parameters as required.

Parameter	Description
Grant To	The accounts to which you want to grant permissions on the project. You can grant permissions on the project to either or both of the personal account and project production account. If you select an account from the **Personal Account** drop-down list, you must also set the **Expiration Date** parameter.
Physical Tables	The names of the physical tables on which you want to grant permissions to the specified accounts. You can select physical tables only in the current project.
Permission Type	The permissions to be granted. Valid values: ○ **Query** ○ **Write** ○ **Update** ○ **Delete**
Reason for Grant of Permissions	The reason why you grant the permissions.

7. Click **OK**.

Revoke permissions on a project from a specific account

1. On the **Project Permissions** tab, find the project on which you want to revoke permissions from a specific account and click the icon in the **Actions** column.
2. In the dialog box that appears, set the parameters as required.

Parameter	Description
Revoke From	The account from which you want to revoke permissions on the project.
Physical Tables	The names of the physical tables on which you want to revoke permissions from the specified account. You can select physical tables only in the current project.
Permission Type	The permissions that you want to revoke.

Parameter	Description
Reason for Revocation of Permissions	The reason why you revoke the permissions.

3. Click **OK**.

9.14.3.2.3. Grant and revoke permissions on data sources

This topic describes how to grant and revoke permissions on data sources.

Grant permissions on a data source to specific accounts

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Security** in the top navigation bar.
4. On the **Permissions** page, click **Data Source Permissions** under **Managed Permissions** in the left-side navigation pane.
5. On the **Data Source Permissions** tab, find the data source on which you want to grant permissions to specific accounts and click the icon in the **Actions** column.
6. In the dialog box that appears, set the parameters as required.

Parameter	Description
Grant To	The accounts to which you want to grant permissions on the data source. You can grant permissions on the data source to either or both of the personal account and project production account. If you select an account from the **Personal Account** drop-down list, you must also set the **Expiration Date** parameter.
Permission Type	The permissions to be granted. Valid values: ○ **Read** ○ **Write**
Reason for Grant of Permissions	The reason why you grant the permissions.

7. Click **OK**.

Revoke permissions on a data source from a specific account

1. On the **Data Source Permissions** tab, find the data source on which you want to revoke permissions from a specific account and click the icon in the **Actions** column.
2. In the dialog box that appears, set the parameters as required.

Parameter	Description
Revoke From	The account from which you want to revoke permissions on the data source.
Permission Type	The permissions that you want to revoke.
Reason for Revocation of Permissions	The reason why you revoke the permissions.

3. Click **OK**.

9.14.3.2.4. Grant and revoke functionality permissions

This topic describes how to grant and revoke functionality permissions.

Grant a functionality permission to a specific account

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Security** in the top navigation bar.
4. On the **Permissions** page, click **Functionality Permissions** under **Managed Permissions** in the left-side navigation pane.
5. On the **Functionality Permissions** tab, find the functionality on which you want to grant the permission to a specific account and click the icon in the **Actions** column.
6. In the dialog box that appears, set the parameters as required.

Parameter	Description
Grant To	The account to which your want to grant the functionality permission.
Expiration Date	The period in which the functionality permission is valid for the specified account.
Reason for Grant of Permissions	The reason why you grant the functionality permission.

7. Click **OK**.

Revoke a functionality permission from a specific account

1. On the **Functionality Permissions** tab, find the functionality on which you want to revoke the permission from a specific account and click the icon in the **Actions** column.
2. In the dialog box that appears, set the parameters as required.

Parameter	Description

Parameter	Description
Revoke From	The account from which you want to revoke the functionality permission.
Reason for Revocation of Permissions	The reason why you revoke the functionality permission.

3. Click **OK**.

9.14.3.3. Owned permissions

9.14.3.3.1. Configure functionality permissions and transfer the ownership of functionality permissions

This topic describes how to configure functionality permissions and transfer the ownership of functionality permissions.

Configure a functionality permission

1. Log on to the Dataphin console.
2. Click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Security** in the top navigation bar.
4. On the **Permissions** page, click **Functionality Permission Settings** under Owned Permissions in the left-side navigation pane.
5. On the **Functionality Permission Settings** tab, find the functionality for which you want to configure the permission and click the icon in the **Actions** column.
6. In the **Configure Permission Management** dialog box, specify the user to approve applications for this permission and specify the maximum validity period of the permission.
7. Click **OK**.

Transfer the ownership of a functionality permission

1. On the **Functionality Permission Settings** tab, find the functionality on which you want to transfer the ownership of the permission and click the icon in the **Actions** column.
2. In the **Transfer Permission Ownership** dialog box, specify the account to which you want to transfer the ownership of the functionality permission.
3. Click **OK**.

9.14.4. Data governance

9.14.4.1. Overview

On the Data Governance page of the Dataphin console, you can analyze resource consumption, control the overall computing and storage costs, and improve service efficiency.

To go to the Data Governance page, perform the following steps:

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Administration** in the top navigation bar. The Data Governance page appears.

The Data Governance page contains the following menus: **Resource Management**, **Governance Overview**, **Governance Console**, **Artifact Management**, and **Recycle Bin**.

- **Resource Management**: includes the **Resource Analysis** page. On the Resource Analysis page, you can view the trends of resource consumption, distribution of resource consumption, and overview of resource governance from a global perspective.
- **Governance Overview**: includes the **Analysis** and **Results** pages. On the **Analysis** page, you can view the diagnosis result of projects for the current tenant and the analysis result of artifacts to be optimized. On the **Results** page, you can view the analysis result of project governance performance. Then, you can evaluate the governance effect and promote the optimization of artifacts.
- **Governance Console**: includes the **My Governance** and **Project Governance** pages. On the **My Governance** page, you can manage and optimize tables or nodes in projects that are owned by the current account. On the **Project Governance** page, you can view the governance information about tables or nodes in projects in which the current account is involved as a member. You can also manage and optimize tables or nodes in projects that are owned by the current account.
- **Artifact Management**: includes the **Metadata Registration**, **Artifact Management**, **Push Management**, and **Task Management** pages. On these pages, you can create artifacts, use custom or system built-in artifacts to configure and initiate push tasks, view the running details and logs of push tasks and artifact tasks, and manage these tasks.
- **Recycle Bin**: allows you to view tables that you have deleted or unpublished in the governance console. Dataphin stores these tables temporarily to prevent misoperation on data.

9.14.4.2. Resource analysis

On the Resource Analysis page, you can view the trends of resource consumption, distribution of resource consumption, and overview of resource governance from a global perspective. This topic describes how to view the global metrics, health status of resources, trend analysis, and project analysis on the Resource Analysis page.

Go to the Resource Analysis page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Resource Analysis** under **Resource Management** in the left-side navigation pane.
5. On the **Resource Analysis** page, you can view the health status of resources, trend analysis, and project analysis.

Global metrics

Global metrics include the **Tables from Services**, **Instances**, **Data Tables**, **Projects**, **Developers**, and **Tables from Computing Engine** parameters.

Parameter	Description
Tables from Services	The number of tables that are synchronized from data sources of specific services to computing engines.
Instances	The total number of task instances with the specified data timestamp.
Data Tables	The total number of tables for the current tenant, including logical tables and physical tables of projects in the production environment and development environment.
Projects	The total number of projects for the current tenant, including projects in the production environment and development environment.
Developers	The total number of Dataphin users who have joined one or more projects of the tenant.
Tables from Computing Engine	The number of tables that are synchronized from computing engines to data sources of specific services.

Resource health status

You can view the health status of both computing resources and storage resources.

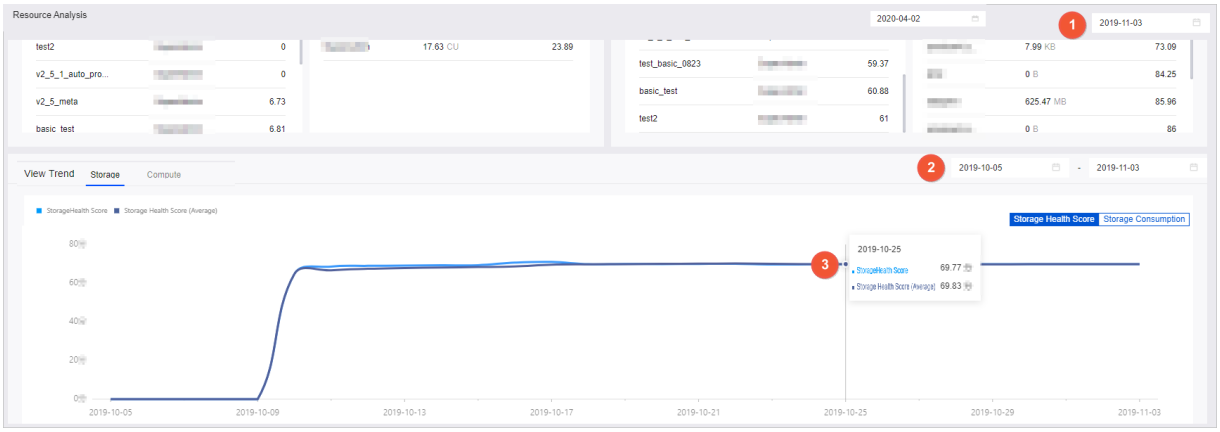
Parameter	Description
Compute Health Score	The score for the health status of computing resources with the specified data timestamp. The health scores are graded into the following three levels: • **High**: The health score is greater than or equal to 85. The message **Your score indicates that you excel in data asset management. Keep up the good work** is displayed. • **Medium**: The health score is greater than or equal to 60 and less than 85. The message **Your score is slightly higher than the threshold. We recommend that you continue to optimize data asset management** is displayed. • **Low**: The health score is less than 60. The message **Your score is lower than the threshold. We recommend that you optimize data asset management at your earliest opportunity** is displayed.

Parameter	Description
Consumed Compute Resources	The amount of consumed computing resources. Unit: CU, KCU, or CM. • 1 core × 60 seconds = 1 CM • 24 hours × 60 minutes × 1 CM = 1 CU • 1,000 CU = 1 KCU Different units are used in the following situations: • If the amount of consumed computing resources is less than 1,000 CM, the unit CM is used, for example, 987 CM. • If the amount of consumed computing resources is greater than 1,000 CM, the unit CU is used, for example, 0.83 CU, which equals to 1,200 CM. • If the amount of consumed computing resources is greater than 1,000 CU, the unit KCU is used, for example, 1.1 KCU, which equals to 1,100 CU. Note • The value is accurate to two decimal places. You can convert the consumption value to a new one with a larger unit as needed. • The amount of consumed computing resources is calculated based on the total execution time for which the CPU was dedicated to underlying tasks. For example, if a task occupies one CPU core and runs for one day, one CU is consumed.
Storage Health Score	The score for the health status of storage resources with the specified data timestamp.
Storage Consumption	The amount of consumed storage resources. Unit: byte, KB, MB, GB, TB, PB, EB, or ZB. Note - The following formulas describe the conversion from the unit of byte to the units of KB, MB, GB, TB, PB, EB, and ZB for the measurement of consumed storage resources: - 1024 B => 1 KB - 1024 * 1024 B => 1 MB - 1024 * 1024 * 1024 B => 1 GB - 1024 * 1024 * 1024 * 1024 B => 1 TB - 1024 * 1024 * 1024 * 1024 * 1024 B => 1 PB - 1024 * 1024 * 1024 * 1024 * 1024 * 1024 B => 1 EB - 1024 * 1024 * 1024 * 1024 * 1024 * 1024 * 1024 B => 1 ZB

Parameter	Description
Projects (Instant Optimization)	The top 10 projects with the lowest health scores in computing or those with the lowest health scores in storage.
Owners (Instant Optimization)	The top 10 owners with the lowest health scores in computing or those with the lowest health scores in storage.

Trend analysis

In the View Trend section, you can view information about global computing resources and storage resources in a specific time period, including the trends of the health score in storage, storage resource consumption, health score in computing, and computing resource consumption. The trend charts help you analyze data stability.



The base date, marked as 1 in the preceding figure, is displayed in the upper-right corner of the page. By default, the trend chart shows the trends of the corresponding data in the 30 days prior to the base date.

- Click the **Storage** tab in the **View Trend** section to view the trend chart of the health score in storage. By default, the trend chart shows the trends of the health score in storage and the average score in the 30 days prior to the base date.

  Move the pointer over a point of time in the chart. You can view the health score in storage corresponding to the specified date and the average score in the seven days prior to the specified date. The data stability is determined based on the difference between the health score in storage and the average score. A smaller difference indicates a higher data stability. For example, at the time point marked as 3 in the preceding figure, the health score in storage of October 25, 2019 is 69.77 and the average score in the seven days prior to October 25, 2019 is 69.83.

You can specify a time period, marked as 2 in the preceding figure, based on business requirements to view the corresponding trends of the health score in storage and the average score. For example, you can specify a time period from October 10, 2019 to October 15, 2019. The trend chart shows the trends of the health score in storage and the average score in this time period, as shown in the following figure. If you move the pointer over the time point of October 14, 2019 in the chart, the corresponding scores appear, that is, 69.23 for the health score in storage on October 14, 2019 and 68.12 for the average score in the seven days prior to October 14, 2019.

- On the Storage tab in the View Trend section, click **Storage Consumption** in the upper-right corner. By default, the trend chart shows the trends of storage resource consumption and the average consumption in the 30 days prior to the base date.

  > **Note** The data on the Storage Consumption chart is presented in the same way as that on the **Storage Health Score** chart.

- Click the **Compute** tab in the **View Trend** section to view the trend chart of the health score in computing. By default, the trend chart shows the trends of the health score in computing and the average score in the 30 days prior to the base date.

  > **Note** The data on the Compute Health Score chart is presented in the same way as that on the **Storage Health Score** chart.

- On the Compute tab in the View Trend section, click **Consumed Compute Resources** in the upper-right corner. By default, the trend chart shows the trends of computing resource consumption and the average consumption in the 30 days prior to the base date.

  > **Note** The data on the Consumed Compute Resources chart is presented in the same way as that on the **Storage Health Score** chart.

Project analysis

In the **Project Analysis** section, you can view the analysis result of projects. You can also filter projects to find a desired project and view the analysis result of the project. View the analysis result of projects in the following ways:

- View the analysis results of the projects that are displayed by default, as described in the following table.

Parameter	Description
Project Name	The name of the project.
Business Unit	The business unit to which the project belongs.
Instances	The number of task instances in the project, including instances of sync tasks and code tasks.
Data Tables	The number of tables in the project, including physical tables and logical tables.
Consumed Compute Resources	The amount of computing resources that are consumed by all tasks in the project. Unit: CU.
Consumed Storage Resources	The amount of storage resources that are consumed by all tables in the project.
Compute Health Score	The health score in computing of the project.
Storage Health Score	The health score in storage of the project.
Administrator	The administrator of the project.
Artifacts	The number of governance items to be optimized.

- Enter a keyword in the search box to search for projects whose names contain the keyword. You can also click the icon and set the **Environment** and **Business Unit** parameters to find desired projects and view the analysis result of these projects.

9.14.4.3. Governance overview

9.14.4.3.1. Governance analysis

On the Analysis page, you can view the diagnosis result of projects for the current tenant and the analysis result of artifacts to be optimized. This topic describes the information you can view on the Analysis page.

Go to the Analysis page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Analysis** under **Governance Overview** in the left-side navigation pane.
5. On the **Analysis** page, click the **Storage** and **Compute** tabs to view the health scores in storage and computing, score details, and project governance details.

   In the upper-right corner of the Analysis page, specify a base date to display the corresponding business data.

Resource categories

By default, the Analysis page displays the health score in storage and score details of projects for the current tenant. You can click the **Compute** tab on the **Analysis** page to view the health score in computing and score details. You can view the following information on the Storage or Compute tab:

- Click the **Storage** tab on the **Analysis** page. The health score in storage appears on the left of the page. The **Score Details** section on the right of the page displays the numbers of artifacts, optimized governance items, and tables to be optimized. The Score Details section also lists information about each artifact, as described in the following table.

Parameter	Description
Artifact Name	The name of the artifact. Artifacts for storage include Excessively Long Retention Period, No Management Policies Specified, Idle Physical Tables, Empty Table, No Management Policies Specified for Logical Tables, Idle Logical Tables, Empty Logical Table, and Logical Table Conversion (Empty Physical Tables).
Score Point Reduction	The points deducted for the occurrence of the artifact.
Storage Resources (To Be Optimized)	The amount of storage resources to be optimized.
Tables (To Be Optimized)	The number of tables to be optimized.

- Click the **Compute** tab on the **Analysis** page.**The health score in computing appears on the left of the page. The Score Details section on the right of the page displays the numbers of artifacts, optimized governance items, and nodes to be optimized.** The Score Details section also lists information about each artifact, as described in the following table.

Parameter	Description
Artifact Name	The name of the artifact. Artifacts for computing include Unread Output Table, Inappropriate Data Scanning, Unexpected Large Amounts of Data, Data Skew, Failed Task Node, No Imported Data, and No Input Data.
Score Point Reduction	The points deducted for the occurrence of the artifact based on the built-in health assessment model.
Compute Resources (To Be Optimized)	The amount of computing resources to be optimized.
Nodes (To Be Optimized)	The number of nodes to be optimized.

Project governance details

In the Project Governance Details section of the Storage and Compute tabs, you can view the governance information about each project in the corresponding business unit, such as the amount of consumed resources, the ratio of optimized governance items to total governance items, and the number of governance items to be optimized. You can click the **Compute** or **Storage** tab on the **Analysis** page to view the governance information about the computing resources or storage resources of projects. You can also filter projects to find a desired project and view the governance details of the project. View the governance details of projects in the following ways:

- Click the **Storage** tab to view the governance information about the storage resources of projects, as described in the following table.

Parameter	Description
Project Name	The name of the project.
Business Unit	The business unit to which the project belongs.
Administrator	The administrator of the project.
Consumed Storage Resources	The amount of storage resources that are consumed by all tables in the project.
Rate of Optimized Storage Resources	The ratio of optimized governance items to total governance items that were identified in the seven days prior to the date you specify in the upper-right corner of the page.
Artifacts	The number of governance items to be optimized in the project.
Actions	The entries for you to view data trends and health score rankings. ◦ Click the icon in the **Actions** column of a project. In the View Trend dialog box, you can view the trend chart of the health score in storage and storage resource consumption in the last 7 or 30 days. You can also move the pointer over a point of time in the chart to view the health score in storage and storage resource consumption of the specified date. ◦ Click the icon in the **Actions** column of a project. In the View Rankings dialog box, you can view the ratio of the storage resources that each table consumes in the project. You can also view the rankings of tables based on the storage resource consumption in descending order or based on the health score in storage in ascending order.

- Click the **Compute** tab to view the governance information about the computing resources of projects, as described in the following table.

Parameter	Description
Project Name	The name of the project.
Business Unit	The business unit to which the project belongs.
Administrator	The administrator of the project.

Parameter	Description
Consumed Compute Resources	The amount of computing resources that are consumed by all nodes in the project.
Rate of Optimized Compute Resources	The ratio of optimized governance items to total governance items that were identified in the seven days prior to the date you specify in the upper-right corner of the page.
Artifacts	The number of governance items to be optimized in the project.
Actions	The entries for you to view data trends and health score rankings. ○ Click the icon in the **Actions** column of a project. In the View Trend dialog box, you can view the trend chart of the health score in computing and computing resource consumption in the last 7 or 30 days. You can also move the pointer over a point of time in the chart to view the health score in computing and computing resource consumption of the specified date. ○ Click the icon in the **Actions** column of a project. In the View Rankings dialog box, you can view the ratio of the computing resources that each node consumes in the project. You can also view the rankings of nodes based on the computing resource consumption in descending order or based on the health score in computing in ascending order.

- Enter a keyword in the search box to search for projects whose names contain the keyword. You can also click the icon and set **Environment**, **Business Unit**, and **Owner** to find desired projects and view the governance details of these projects.

9.14.4.3.2. Governance results

On the Results page, you can view the analysis result of project governance performance. Then, you can evaluate the governance effect and promote the optimization of artifacts. This topic describes how to view the overall governance metrics, trends, and project governance details on the Results page.

Go to the Results page

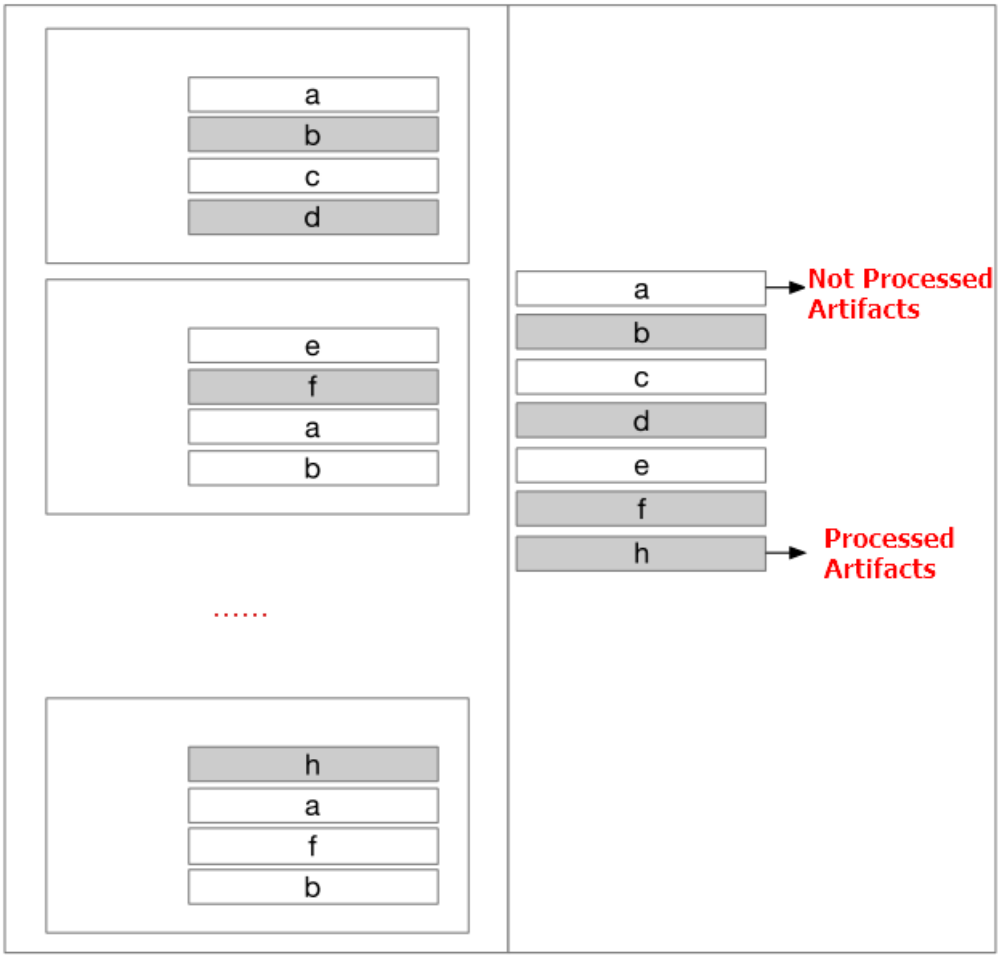
1. Log on to the Dataphin console.
2. Go to the **Data Governance** page.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Administration** in the top navigation bar.
3. On the **Data Governance** page, click **Results** under **Governance Overview** in the left-side navigation pane.

   In the upper-right corner of the **Results** page, specify a base date to display the corresponding business data. The Results page consists of the following sections: **Artifact Details**, **Saved Resources**, **View Trend**, and **Project Governance Details**.

Overall governance metrics

The overall governance metrics include two categories: **Artifact Details** and **Saved Resources**. The **Artifact Details** section displays the **Rate of Processed Resources** and **Processed Artifacts** metrics. The **Saved Resources** section displays the **Compute** and **Storage** metrics.

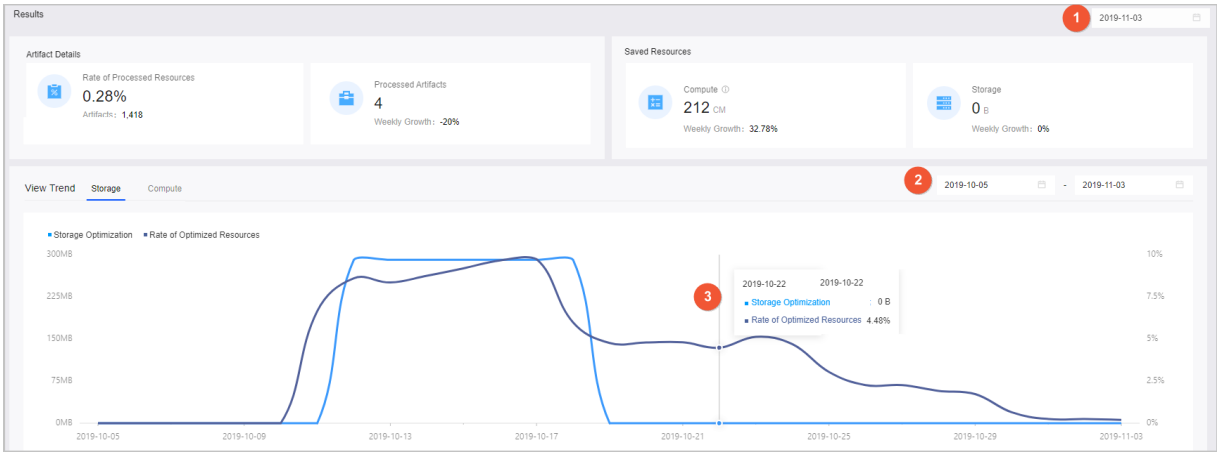
- **Rate of Processed Resources**: the ratio of optimized governance items to total governance items that were identified in the last seven days.



- **Processed Artifacts**: the number of governance items that were optimized in the last seven days.
- **Compute**: the amount of computing resources that have been saved by optimized governance items of the computing type.
- **Storage**: the amount of storage resources that have been saved by optimized governance items of the storage type.

Trend analysis

In the View Trend section, you can view information about global computing resources and storage resources in a specific time period, including the trends of the amount of saved resources and the ratio of optimized governance items to total governance items.

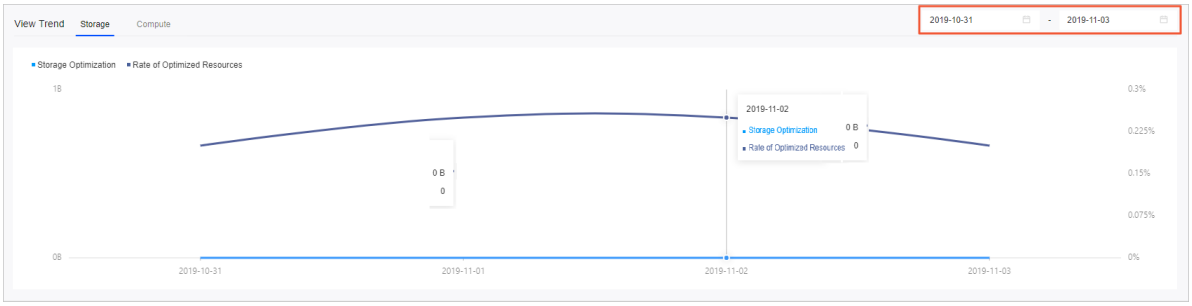


The base date, marked as 1 in the preceding figure, is displayed in the upper-right corner of the page. By default, the trend chart shows the trends of the corresponding data in the 30 days prior to the base date.

- Click the **Storage** tab in the **View Trend** section to view the trends of the amount of saved storage resources and the ratio of optimized governance items of the storage type to total governance items of the storage type in the 30 days prior to the base date.

  Move the pointer over a point of time in the chart. You can view the ratio of optimized governance items of the storage type to total governance items of the storage type corresponding to the specified date and the amount of storage resources that have been saved in the seven days prior to the specified date. For example, at the time point marked as 3 in the preceding figure, the amount of storage resources that have been saved in the seven days prior to October 22, 2019 is 0 bytes and the ratio of optimized governance items of the storage type to total governance items of the storage type on that day is 4.48%.

  You can specify a time period, marked as 2 in the preceding figure, based on business requirements to view the corresponding trends of the ratio of optimized governance items of the storage type to total governance items of the storage type and the amount of saved storage resources. For example, you can specify a time period from October 31, 2019 to November 3, 2019. The trend chart shows the trends of the ratio of optimized governance items of the storage type to total governance items of the storage type and the amount of saved storage resources in this time period, as shown in the following figure. If you move the pointer over the time point of November 2, 2019 in the chart, the corresponding data appears, that is, 0.0% for the ratio of optimized governance items of the storage type to total governance items of the storage type on November 2, 2019 and 0 bytes for the amount of storage resources that have been saved in the seven days prior to November 2, 2019.

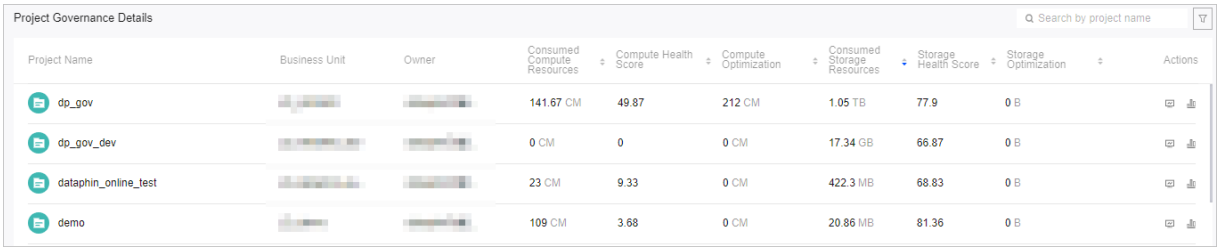


- Click the **Compute** tab in the **View Trend** section to view the trends of the amount of saved computing resources and the ratio of optimized governance items of the computing type to the total governance items of the computing type in the 30 days prior to the base date.

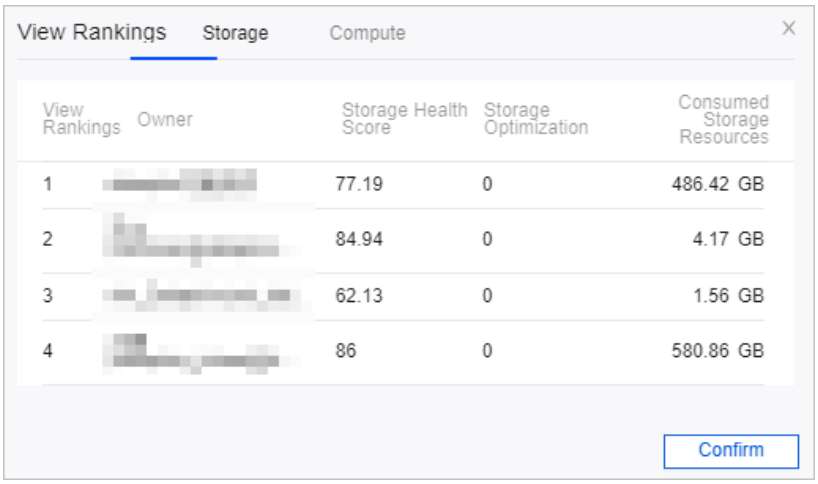
> **Note** The data on the Compute tab is presented in the same way as that on the Storage tab.

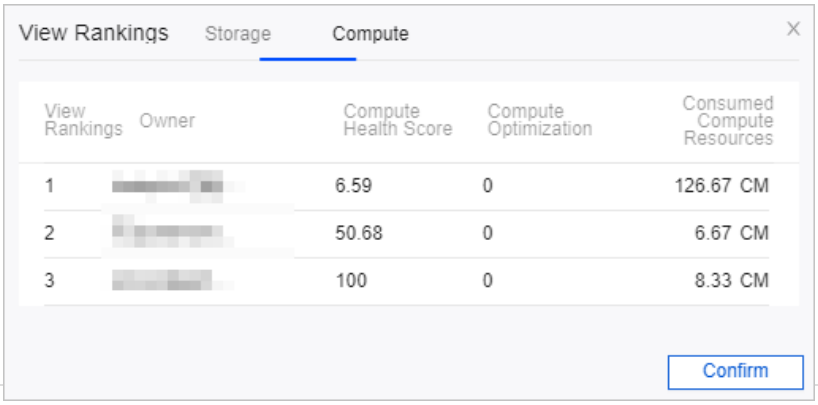
Project governance details

In the **Project Governance Details** section, you can view the governance information about each project in the corresponding business unit, such as the amount of consumed storage or computing resources, the health score in storage or computing, and the amount of saved storage or computing resources.



Parameter	Description
Project Name	The name of the project.
Business Unit	The business unit to which the project belongs.
Owner	The owner of the project.
Consumed Storage Resources	The amount of storage resources that are consumed by all tables in the project.
Storage Health Score	The health score in storage of the project.
Consumed Compute Resources	The amount of computing resources that are consumed to run MaxCompute tasks in the project, including SQL tasks that are created by developers and automatically generated by Dataphin.
Compute Health Score	The health score in computing of the project.

Parameter	Description
Storage Optimization	The amount of storage resources that have been saved in the project by optimized governance items of the storage type.
Compute Optimization	The amount of computing resources that have been saved in the project by optimized governance items of the computing type.
Actions	The entries for you to view data trends and health score rankings. • Click the icon in the **Actions** column of a project. In the View Trend dialog box, you can view the trend chart of the corresponding data in the last 7 or 30 days. ○ On the **Storage** tab, move the pointer over a point of time in the chart. You can view the health score in storage and storage resource consumption of the specified date. ○ Click the **Compute** tab. On the **Compute** tab, move the pointer over a point of time in the chart. You can view the health score in computing and computing resource consumption of the specified date. • Click the icon in the **Actions** column of a project. In the View Rankings dialog box, you can view the rankings of project owners based on their project governance performance. ○ On the **Storage** tab, you can view the rankings of project owners based on the health score in storage, amount of consumed storage resources, and amount of saved storage resources. 

Parameter	Description
	○ Click the **Compute** tab. On the Compute tab, you can view the rankings of project owners based on the health score in computing, amount of consumed computing resources, and amount of saved computing resources. 

Enter a keyword in the search box to search for projects whose names contain the keyword. You can also click the icon and set **Environment**, **Business Unit**, and **Owner** to find desired projects and view the governance details of these projects.

9.14.4.4. Governance console

In the governance console, you can view the governance information about tables or nodes in projects in which you are involved as a member. You can also manage and optimize tables or nodes in projects in which you are involved as an administrator.

As the administrator or developer of a project, you can filter tables or nodes that require optimization by selecting the governance scope, governance target, and artifact, and then view corresponding information and manage and optimize governance items in the governance console. The features provided on the **My Governance** page are similar to those on the **Project Governance** page. The following example demonstrates how to use the governance console. The **My Governance** page is used in this example.

Go to the My Governance page

1. Log on to the Dataphin console.
2. Go to the **My Governance** page.
   i. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
   ii. On the **Data Assets** page, click **Administration** in the top navigation bar.
   iii. On the **Data Governance** page, click **My Governance** under **Governance Console** in the left-side navigation pane.

iv. On the **My Governance** page, set the **Governance Scope**, **Governance Target**, and **Artifact** parameters to filter governance items.

Parameter	Description
Governance Scope	The business scope of the governance target. Valid values: **Storage** and **Compute**.
Governance Target	■ If you select **Storage** for the **Governance Scope** parameter, the valid values of the **Governance Target** parameter are **Physical Table** and **Logical Table**. ■ If you select **Compute** for the **Governance Scope** parameter, the value of the **Governance Target** parameter is fixed to **Task Node**.
Artifact	■ If you select **Storage** for the **Governance Scope** parameter and select **Physical Table** for the **Governance Target** parameter, the valid values of the **Artifact** parameter are **No Management Policies Specified**, **Empty Table**, **Idle Physical Tables**, **Excessively Long Retention Period**, and custom artifacts. ■ If you select **Storage** for the **Governance Scope** parameter and select **Logical Table** for the **Governance Target** parameter, the valid values of the **Artifact** parameter are **No Management Policies Specified for Logical Tables**, **Idle Logical Tables**, **Empty Logical Table**, **Logical Table Conversion (Empty Physical Tables)**, and custom artifacts. ■ If you select **Compute** for the **Governance Scope** parameter and select **Task Node** for the **Governance Target** parameter, the valid values of the **Artifact** parameter are **No Input Data**, **No Imported Data**, **Failed Task Node**, **Data Skew**, **Unexpected Large Amounts of Data**, **Inappropriate Data Scanning**, and **Unread Output Table**.

v. View the details of governance items.

- If you select **Storage** for the **Governance Scope** parameter and select **Physical Table** for the **Governance Target** parameter, you can view information about each physical table to be optimized.

Parameter	Description
Table Name	The name of the physical table.
Project Name	The name of the project to which the physical table belongs.
Business Unit	The business unit to which the physical table belongs.
Owner	The owner of the physical table.
Health Score	The health score of the physical table.
Storage Space	The size of MaxCompute storage resources that the physical table occupies.
Retention Period	The retention period of the physical table.
Number of Days (Accessed from First Day to Last Day in the Last 33 Days)	The number of visits to the physical table in the last 33 days.
Processing Status	The processing status of the node. Valid values: **Processed** and **Not Processed**.

- If you select **Storage** for the **Governance Scope** parameter and select **Logical Table** for the **Governance Target** parameter, you can view information about each logical table to be optimized.

Parameter	Description
Model Name	The name of the logical table.
Project Name	The name of the project to which the logical table belongs.
Business Unit	The business unit to which the logical table belongs.
Owner	The owner of the logical table.
Health Score	The health score of the logical table.
Storage Space	The size of MaxCompute storage resources that the logical table occupies.
Retention Period	The retention period of the logical table.
Environment	The environment to which the logical table belongs. Valid values: **Development** and **Production**.
Processing Status	The processing status of the node. Valid values: **Processed** and **Not Processed**.

- If you select **Compute** for the **Governance Scope** parameter and select **Task Node** for the **Governance Target** parameter, you can view information about each node to be optimized.

Parameter	Description
Node Name	The name of the node.
Node ID	The ID of the node.
Node Type	The type of the node.
Project Name	The name of the project to which the node belongs.
Owner	The owner of the node.
Health Score	The health score of the node.
CPU Consumption	The amount of computing resources that the node consumes.
Duration	The duration during which the node instance is run.
Processing Status	The processing status of the node. Valid values: **Processed** and **Not Processed**.

Manage physical tables

The supported actions on physical tables include **Move to Recycle Bin**, **Set Retention Period**, and **Pause Node**.

1. On the **My Governance** page, you can find physical tables to be optimized by using one of the following methods:
   - Select **Storage** for the **Governance Scope** parameter and select **Physical Table** for the **Governance Target** parameter to filter physical tables.
   - Click the icon next to **Select a project** and select a project to filter physical tables.
   - Enter a keyword in the search box to search for physical tables whose names contain the keyword.
   - Click the icon next to **All** and select All, Processed, or Not Processed to filter physical tables.
2. Manage physical tables by using one of the following methods:
   - Find the physical table that you want to manage and click the icon in the **Actions** column to move the physical table to the recycle bin.

     You can also select multiple physical tables and click **Move to Recycle Bin** in the lower part of the page to move them to the recycle bin. In the **Confirm** message, click **Confirm**. The selected physical tables are moved to the recycle bin.
   - Find the physical table that you want to manage and click the icon in the **Actions** column. In the **Set Retention Period** dialog box, set a retention period and click **OK**.

     You can also select multiple physical tables and click **Set Retention Period** in the lower part of the page. In the **Set Retention Period** dialog box, set a retention period and click **OK**. A retention period is set for the selected physical tables.
   - Find the physical table that you want to manage and click the icon in the **Actions** column. In the **Confirm** message, click **Confirm**. The system stops using this physical table.

     > **Note** The system does not allow you to resume a paused physical table.

Manage logical tables

Only the **Set Retention Period** action is supported on logical tables.

1. On the **My Governance** page, you can find logical tables to be optimized by using one of the following methods:
   - Select **Storage** for the **Governance Scope** parameter and select **Logical Table** for the **Governance Target** parameter to filter logical tables.
   - Click the icon next to **Select a project** and select a project to filter logical tables.
   - Enter a keyword in the search box to search for logical tables whose names contain the keyword.
   - Click the icon next to **All** and select All, Processed, or Not Processed to filter logical tables.
2. Find the logical table that you want to manage and click the icon in the **Actions** column. In

the **Set Retention Period** dialog box, set a retention period and click **OK**.

You can also select multiple logical tables and click **Set Retention Period** in the lower part of the page. In the **Set Retention Period** dialog box, set a retention period and click **OK**. A retention period is set for the selected logical tables.

Manage nodes

The supported actions on nodes include **Pause Node** and **Unpublish Node**.

1. On the **My Governance** page, you can find nodes to be optimized by using one of the following methods:
   - Select **Compute** for the **Governance Scope** parameter and select **Task Node** for the **Governance Target** parameter to filter nodes.
   - Click the icon next to **Select a project** to filter nodes.
   - Enter a keyword in the search box to search for nodes whose names contain the keyword.
   - Click the icon next to **All** and select All, Processed, or Not Processed to filter nodes.
2. Manage nodes by using one of the following methods:
   - Find the node that you want to manage and click the icon in the **Actions** column. In the **Confirm** message, click **Confirm**. The system stops running this node.

     > **Note** The system does not allow you to resume a paused node.

   - Find the node that you want to manage and click the icon in the **Actions** column. In the **Confirm** message, click **Confirm**. The node is not run in the production environment as scheduled.

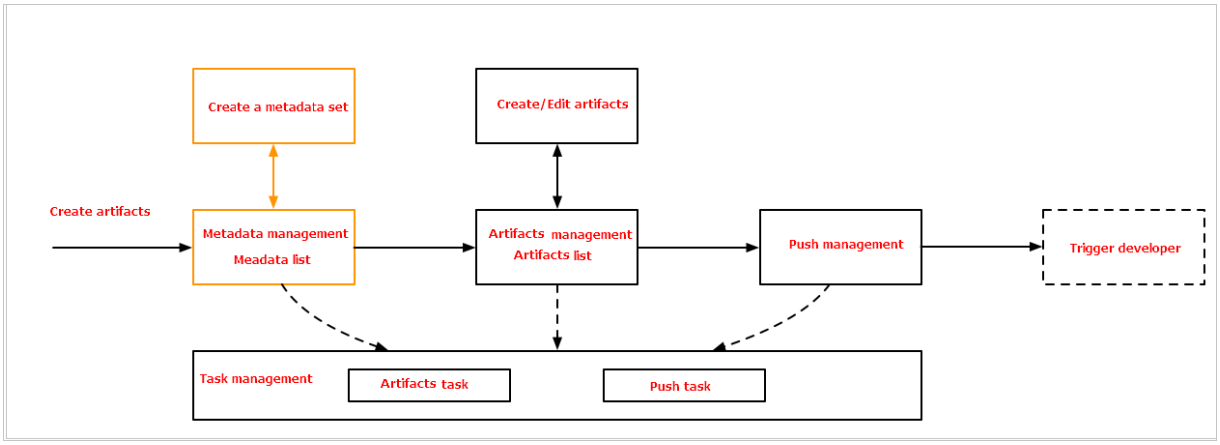
9.14.4.5. Artifact management

9.14.4.5.1. Overview

In the Artifact Management section, you can create and manage metadata sets, artifacts, and push tasks, view the running details and logs of push tasks and diagnosis tasks, and manage these tasks. In the Artifact Management section, you can create and manage custom artifacts and manage system built-in artifacts.

Artifact management Metadata management Push management Task management

Dataphin allows you to create custom artifacts based on your business requirements and use the governance settings in data asset management.



The following features are provided by Dataphin:

1. Metadata management: allows you to create and manage metadata sets required for data governance.
2. Artifact management: allows you to create and manage artifacts based on the configured metadata sets. Artifacts define the identification rules and policies for data governance.
3. Push management: allows you to create push tasks so that you can receive push notifications when occurrences of one or more specified artifacts are identified for the data objects, including tables and nodes, in the selected projects. The push notifications help to remind you to process the data objects that require optimization.
4. Task management: allows you to manage diagnosis tasks and push tasks.

9.14.4.5.2. Metadata registration

On the Metadata Registration page, you can create and manage metadata sets that are required for data governance. This topic describes how to create, edit, check, and delete metadata sets, and change owners of metadata sets.

Go to the Metadata Registration page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Metadata Registration** under **Artifact Management** in the left-side navigation pane.
5. On the **Metadata Registration** page, view information about registered metadata sets.

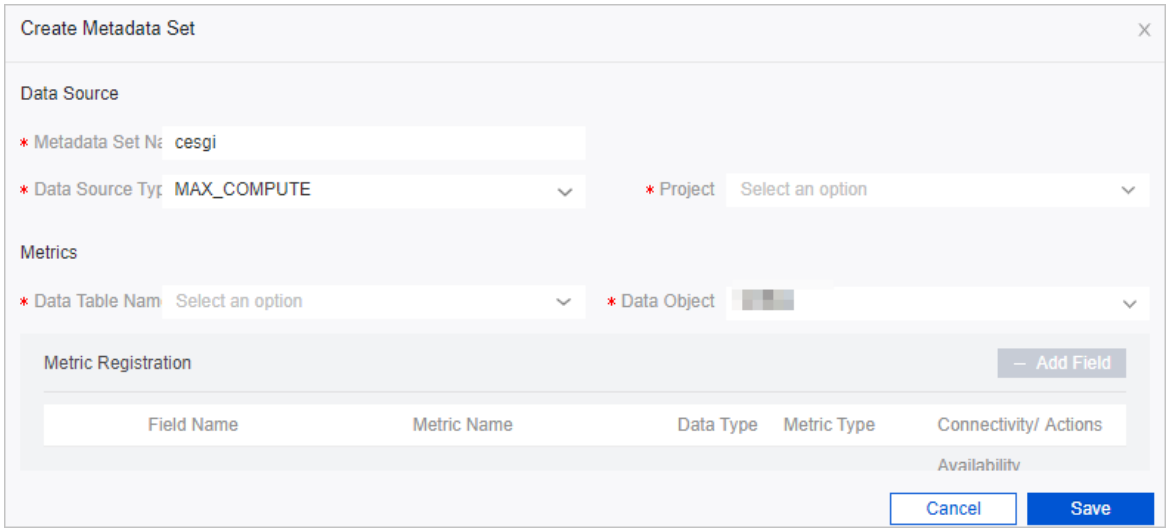
Parameter	Description
Metadata Set Name	The name of the metadata set.
Data Source Type	The type of the data source for the metadata set.
Data Object	The type of the data object that is managed based on the metadata set. Valid values: **Physical Table**, **Logical Table**, and **Task Node**.

Parameter	Description
Owner	The owner of the metadata set.
Metrics	The number of metrics that are registered in the metadata set.
Referenced by Artifacts	Indicates whether the metadata set is referenced and the number of times that the metadata set is referenced. For example, the value is Yes if you reference a metric in the metadata set when you create an artifact.
Check Status	The check result of the metadata set. Valid values: **Succeeded** and **Failed**. If any fields that the metadata set references are missing or the referenced table is renamed, the check fails. If all fields that the metadata set references exist and the metadata set can properly connect to the referenced table, the check is successful.

You can enter a keyword in the search box to search for metadata sets whose names contain the keyword. You can also click the icon and set the **Data Object**, **Data Source Type**, **Referenced Status**, and **Owner** parameters to filter metadata sets.

Create a metadata set

1. On the **Metadata Registration** page, click **Create Metadata Set** in the upper-right corner.
2. In the **Create Metadata Set** dialog box, set the parameters as required.



Section	Parameter	Description
	Metadata Set Name	The name of the metadata set. Limits: ○ The name must be 1 to 30 characters in length. ○ The name must be unique.

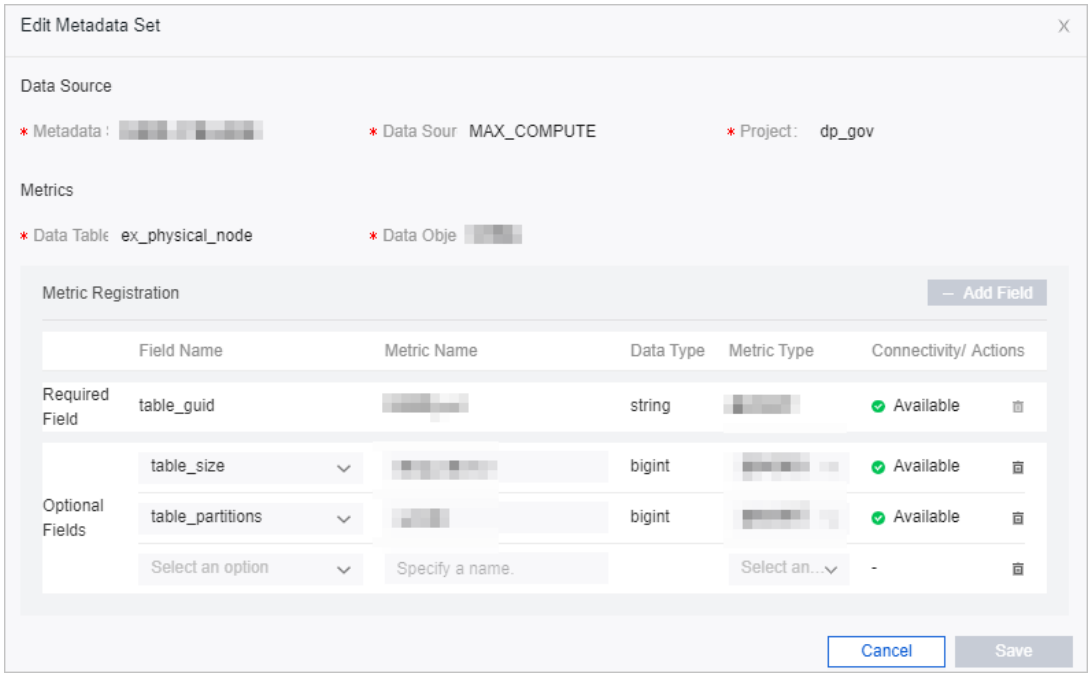
Section	Parameter	Description
Data Source	Data Source Type	The type of the data source for the metadata set. Select **MAX_COMPUTE**.
	Project	The data source of the metadata set. Select a computing engine from the drop-down list or search for it by entering a keyword in the search box. **Note** The drop-down list displays the computing engines that are created on the Planning page of the Dataphin console.
Metrics	Data Table Name	The table that is referenced by the metadata set. Select a table from the drop-down list or search for it by entering a keyword in the search box. **Note** Limits: ○ The drop-down list displays the physical tables, logical tables, and nodes that belong to the project of the selected computing engine. ○ You can create only one metadata set based on each table.
	Data Object	The type of the data object to be managed based on the metadata set. Valid values: **Physical Table**, **Logical Table**, and **Task Node**.
	Metric Registration	The metrics that are registered in the metadata set. At least one field of each table must be registered in the metadata set. You can click **Add Field** to add more fields. For more information, see Edit a metadata set.

3. Click **Save**. The metadata set is created.

Edit a metadata set

1. On the **Metadata Registration** page, find the metadata set that you want to edit and click the icon in the **Actions** column.
2. In the **Edit Metadata Set** dialog box, add or remove fields as required. The other parameters cannot be modified.
   - Add fields to the metadata set.

a. Click **Add Field** and set the parameters as required.



Parameter	Description
Field Name	The field to be used as a metric. Select a field from the drop-down list. **Note** The drop-down list displays the fields in the referenced table. Each field can be added only once.
Metric Name	The name of the metric.
Data Type	The data type of the field. Default value: **string**.
Metric Type	The type of the metric. Valid values: **Basic Attributes**, **Basic Metrics**, and **My Metrics**.
Connectivity/Availability	The connection status of the field. A field can be in the Available or Unavailable state. If a field in the **Metric Registration** section is in the Unavailable state, you cannot save the added fields.

b. Click **Save**. The fields are added to the metadata set.

- Delete a field from the metadata set.

  Find the field that you want to delete and click the icon in the **Actions** column to delete the field.

> **Note** Limits:
>
> - You cannot delete the first field in the **Required Field** section.
> - You cannot edit system metadata sets.

3. Click **Save**. The metadata set is modified.

Check a metadata set

On the Metadata Registration page, find the metadata set that you want to check and click the icon in the **Actions** column to check the metadata set.

Change the owner of a metadata set

1. On the Metadata Registration page, find the metadata set for which you want to change the owner and click the icon in the **Actions** column.To change the owners of multiple metadata sets at a time, select the metadata sets and click **Change Owner** in the lower part of the page.
2. In the **Change Owner** dialog box, select the user to whom you want to transfer the metadata set.
3. Click **Change Owner**. The metadata set is transferred to the specified user.

> **Note** Limits:
>
> - After the owner of a metadata set is changed, the original owner cannot modify the metadata set.
> - You can change the owner of a metadata set to any user of the current tenant.
> - You cannot change owners of system metadata sets.

Delete a metadata set

1. On the **Metadata Registration** page, find the metadata set that you want to delete, click the icon, and then select **Delete** in the **Actions** column.
2. In the **Delete Metadata Set** message, click **Confirm**.

> **Note** You cannot delete system metadata sets.

9.14.4.5.3. Artifact management

This topic describes how to create, edit, and delete artifacts.

Context

The following table describes the limits on operations related to artifact management.

Operation	Limit
Create an artifact	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator and project administrator
Edit an artifact	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator, project administrator, and artifact owner
View an artifact	This operation can be performed by all users.
Enable an artifact	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator, project administrator, and artifact owner
Disable an artifact	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator, project administrator, and artifact owner
Change the owner	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator, project administrator, and artifact owner
Delete an artifact	Limits on the role: • System built-in artifact: super administrator only • Custom artifact: super administrator, project administrator, and artifact owner

Create an artifact

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the Data Assets page, click **Administration** in the top navigation bar.
4. On the Data Governance page, click **Artifact Management** under Artifact Management in the left-side navigation pane.
5. On the **Artifact Management** page, click **Create Artifact** in the upper-right corner.
6. On the **Create Artifact** page, set the parameters as required.

i. Select an artifact type.

Artifact type	Description
Custom artifact	The artifact that you can customize based on your business needs.
System built-in artifact	The artifact that can be created by the super administrator by using an Apsara Stack account. Dataphin pushes notifications for system built-in artifacts to all users.

ii. Set the basic parameters for the artifact.

Parameter	Description
Artifact Name	The name of the artifact. Dataphin has the following limits on the artifact name: ■ The name must be 1 to 20 characters in length. ■ The name must be unique.

Parameter	Description
Governance Target	The type of the data object to be managed. Valid values: ■ **Physical Table** After you select this option, display fields and governance operations are selected by default. You can clear some display fields and operations as required. The following display fields and operations are available: ■ Display fields: **Project Name**, **Business Unit**, **Owner**, **Health Score**, **Storage Space**, **Retention Period**, and **Number of Days (Accessed from First Day to Last Day in the Last 33 Days)** ■ Operations: **Move to Recycle Bin**, **Set Retention Period**, and **Pause Node** The selected display fields and operations will appear in the governance console. ■ **Logical Table** After you select this option, display fields and governance operations are selected by default. You can clear some display fields and operations as required. The following display fields and operations are available: ■ Display fields: **Project Name**, **Business Unit**, **Owner**, **Health Score**, **Storage Space**, and **Retention Period** ■ Only the **Set Retention Period** operation is supported. You cannot clear this operation. The selected display fields and operations will appear in the governance console. ■ **Task Node** After you select this option, display fields and governance operations are selected by default. You can clear some display fields and operations as required. The following display fields and operations are available: ■ Display fields: **Node ID**, **Node Type**, **Owner**, **Project Name**, **Health Score**, **CPU Consumption**, and **Duration** ■ Operations: **Pause Node** and **Unpublish Node** The selected display fields and operations will appear in the governance console.
Governance Scope	The business scope of the governance target. This parameter is automatically set based on the selected governance target. Valid values: **Compute** and **Storage**.
Description	The description of the artifact.

iii. Set rules for the artifact.

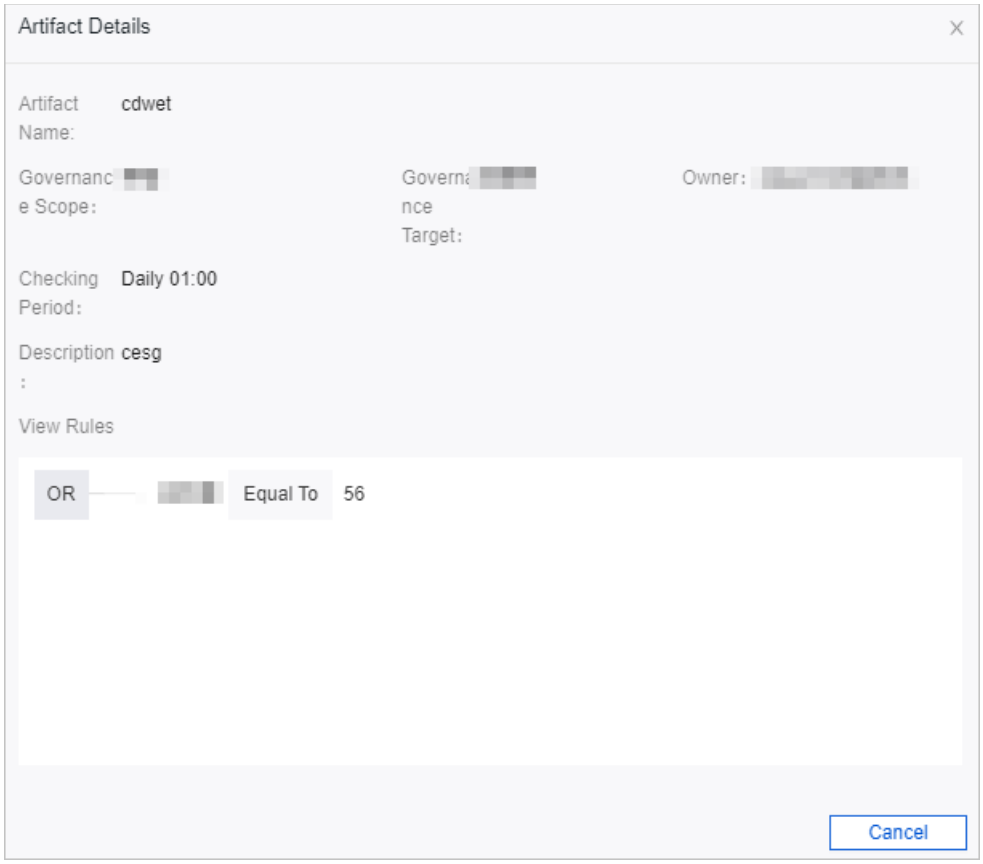
Section	Parameter	Description
Edit Rules	Governance Method	The method of setting rules for the artifact. You can use the codeless UI or code editor. ■ If you set the **Artifact Type** parameter to **Custom Artifact**, you can select **Create Custom Rules** or **Reference System Rules** in the **Configure Rules** section. ■ If you set the **Artifact Type** parameter to **General Artifact**, you can select only **Create Custom Rules** in the **Configure Rules** section.
	Code Editor	The custom SQL statements for setting rules for the artifact. If you set the **Governance Method** parameter to **Code Editor**, enter SQL statements in the code editor. Example: `select` `table_guid` `,business_name` `,is_important` `from` `physical_table_individuation` `where` `ds='${bizdate}'` `and business_name='tmall'` `and is_relation='Y'` `and table_size>100` **Note** This method uses SQL statements to customize rules based on the metrics of registered metadata sets. The artifact uses these rules to manage governance targets.

Section	Parameter	Description
Configure Rules	Create Custom Rules	To add rules, perform the following steps: a. To add an operator, click **Add Operator** and select **OR** or **AND**. b. To add a rule, click **Add Rule**, select a tag and a relational operator, and then enter a value. Parameter description: ■ Tags are registered metrics of metadata sets. ■ Relational operators include Greater Than, Greater Than or Equal To, Equal To, Less Than or Equal To, and Less Than. To remove an operator or a rule, click the icon next to the operator or rule.
	Reference System Rules	To reference the rules of a system built-in artifact, perform the following steps: a. Click **Reference System Rules**. b. In the **Reference System Rules** dialog box, select a system built-in artifact from the drop-down list. Note The system displays details of the selected system built-in artifact. c. Click **Confirm**.

7. Click **Submit**. The artifact is created.

View an artifact

1. On the **Artifact Management** page, find the artifact that you want to view and click the icon in the **Actions** column.
2. In the **Artifact Details** message, view the details of the artifact.



Edit an artifact

1. On the **Artifact Management** page, find the artifact that you want to edit and click the icon in the **Actions** column.
2. On the **Edit Artifact** page, modify the parameters for the artifact. For more information, see Create an artifact.

Enable an artifact

1. On the **Artifact Management** page, find the artifact that you want to enable and click the icon in the **Actions** column.
2. In the **Enable Artifact** message, click **Confirm**.

Disable an artifact

1. On the **Artifact Management** page, find the artifact that you want to disable and click the icon in the **Actions** column.
2. In the **Disable Artifact** message, click **Confirm**.

Change the owner of an artifact

1. On the **Artifact Management** page, find the artifact for which you want to change the owner and click the icon in the **Actions** column.
2. Select **Change Owner**.

3. In the **Change Owner** dialog box, select the user to whom you want to transfer the artifact.
4. Click **Confirm**.

> **Note** After the artifact is transferred to the selected user, the original owner cannot modify the artifact.

Delete an artifact

1. On the **Artifact Management** page, find the artifact that you want to delete and click the icon in the **Actions** column.
2. Select **Delete**.
3. In the **Delete Artifact** message, click **Confirm**.

9.14.4.5.4. Push management

Dataphin pushes governance results to you by using SMS messages, emails, and internal messages. This topic describes how to create, edit, and delete push tasks.

Go to the Push Management page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the Data Assets page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Push Management** under **Artifact Management** in the left-side navigation pane.
5. On the Push Management page, click the **As Owner** or **All** tab to view your push tasks or all push tasks.

Parameter	Description
Topic-based Push Task	The name of the push task.
Owner	The owner of the push task.
Start Date	The start date of the push task.
End Date	The end date of the push task.
Status	The status of the push task. Valid values: ○ **Publish** ○ **Pause** ○ **Unpublish**

You can enter a keyword in the search box to search for push tasks whose names contain the keyword. You can also click the icon and set **Status** to filter push tasks.

Create a push task

1. On the **Push Management** page, click **Create Push Task** in the upper-right corner.
2. On the **Create Push Task** page, set the parameters as prompted.
   i. Set the parameters in the **Push Task Settings** section.

Parameter	Description
Topic-based Push Task	The name of the push task.
Effective Time Range	The period during which the push task is effective.
Notification Time	The time for triggering the push task. ■ If you select **Specified Time**, select a date and the time for pushing governance results. ■ If you select **Recurrence**, select an interval and the time for pushing governance results.
Notification Methods	The methods for pushing governance results. Valid values: **SMS**, **Email**, and **Internal Message**.

   ii. Set the parameters in the **Push Data Settings** section.

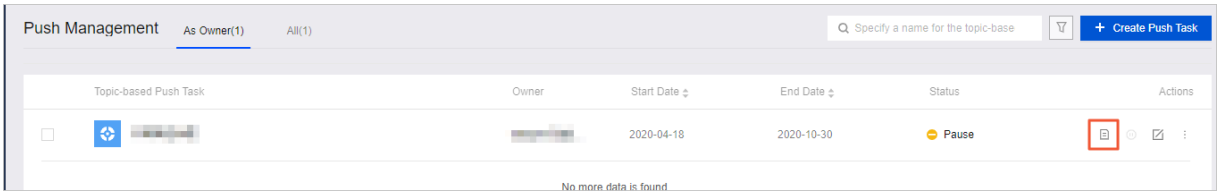
Button	Description
Create Artifact	If the artifacts you want are unavailable, click **Create Artifact** to create ones. For more information, see Create an artifact.
Add Existing Artifacts	To add an existing artifact, perform the following steps: a. Click **Add Existing Artifacts**. b. In the **Add Existing Artifacts** dialog box, find the artifact that you want to add in the **Available Artifacts** section and click the  icon to add it to the **Selected Artifacts** section. You can filter artifacts by using one of the following methods: ■ Enter a keyword in the search box to search for artifacts whose names contain the keyword. ■ Click the icon and set **Type**, **Owner**, and **Governance Target** to filter artifacts. c. Click **OK**.

   iii. Click **Next** in the **Push Data Settings** section.
   iv. Select a project and click **Test**.
3. Wait until the **Sent** message appears on the **Create Push Task** page, and click **Submit**.
4. In the **Submit Confirm** message, click **Confirm Submit**.

> **Note** By default, a new push task enters the Publish state after you submit it

View a push task

On the **Push Management** page, find the push task that you want to view and click the icon in the **Actions** column. In the **View Details** message, you can view the settings and involved artifacts and projects about the push task.



Edit a push task

1. On the **Push Management** page, find the push task that you want to edit and click the icon in the **Actions** column.
2. On the **Edit Push Task** page, set the parameters as prompted. For more information, see Create a push task.
3. Wait until the **Sent** message appears on the **Edit Push Task** page, and click **Submit**.
4. In the **Submit Confirm** message, click **Confirm Submit**.

Pause a push task

1. On the **Push Management** page, find the push task that you want to pause and click the icon in the **Actions** column.
2. In the **Pause Confirm** message, click **Confirm Pause**.

Change the owner of a push task

1. On the **Push Management** page, find the push task for which you want to change the owner, click the icon, and then select **Change Owner** in the **Actions** column.You can also select multiple push tasks and click **Change Owner** in the lower part of the page. In the **Change Owner** dialog box, select the user to whom you want to change the owner of the push tasks.
2. In the **Change Owner** dialog box, select the user to whom you want to transfer the push task.
3. Click **Confirm**.

> **Note**
> - After the owner of a push task is changed, the original owner cannot modify the push task.
> - You can transfer a push task to any user of the current tenant.

Delete a push task

1. On the **Push Management** page, find the push task that you want to delete, click the icon, and then select **Delete** in the **Actions** column.
2. In the **Delete Confirm** message, click **Confirm Delete**.

9.14.4.5.5. Task management

On the Task Management page, you can manage artifact tasks and push tasks. This topic describes how to view task details and task logs and rerun tasks.

Artifact tasks are used as examples in this topic. Follow the same procedures to manage push tasks.

Go to the Task Management page

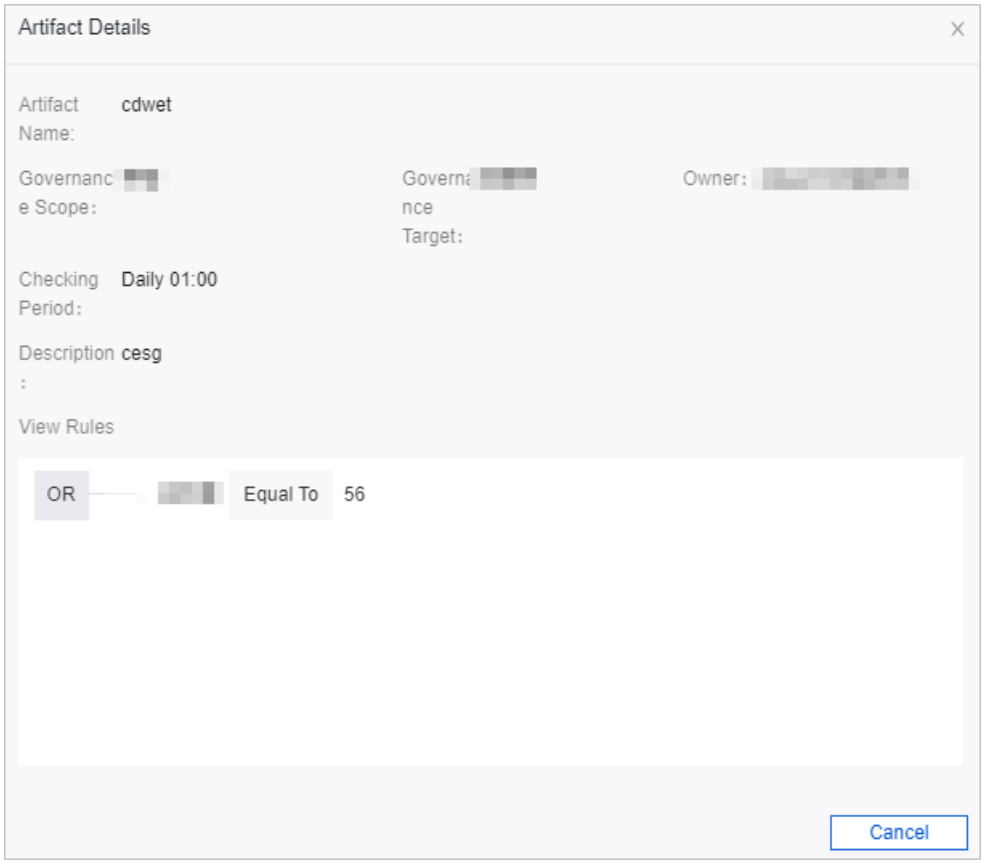
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Task Management** under Artifact Management in the left-side navigation pane.
5. On the **Task Management** page, click the **Push** or **Artifact** tab to view push tasks or artifact tasks.

Parameter	Description
Task Name	The name of the task.
Owner	The owner of the task.
Task Status	The running status of the task. Valid values: ○ **Succeeded** ○ **Failed** ○ **Initializing** ○ **Executing** ○ **Stopped**
Date Timestamp	The date on which the task is scheduled. The value of this parameter is the default value of the bizdate parameter.
Runtime	The period of time during which the task was run.

You can enter a keyword in the search box to search for tasks whose names contain the keyword. You can also click the icon and set **Owner**, **Data Timestamp**, **Runtime**, and **Task Status** to filter tasks.

View the details of a task

On the Task Management page, find the task that you want to view and click the icon in the Actions column. The View Details message appears.



Rerun a task

On the Task Management page, find the task that you want to rerun and click the icon in the Actions column to rerun the task. Then, the task enters the Executing state. After the task is rerun, the task enters the Succeeded state.

Pause a task

On the Task Management page, find the task that you want to pause and click the icon in the Actions column to pause the task.

View the logs of a task

On the Task Management page, find the task whose logs you want to view, click the icon, and then select View Log in the Actions column. The View Task Log message appears.

9.14.4.6. Recycle bin

To prevent misoperations on tables, Dataphin allows you to store deleted tables in the recycle bin for a short period of time. If a table is deleted by mistake, you can restore the table from the recycle bin. If a table is no longer required, you can clear the table from the recycle bin. This topic describes how to clear and restore tables in the recycle bin and view the operational logs of a table.

Go to the Tables page

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the Data Assets page, click **Administration** in the top navigation bar.
4. On the **Data Governance** page, click **Tables** under Recycle Bin in the left-side navigation pane.
5. On the **Tables** page, click the **To Be Processed**, **Processed**, or **All** tab to view pending tables, processed tables, or all tables that are moved to the recycle bin.

Parameter	Description
Table Name	The name of the table.
Project Name	The project to which the table belongs.
Table Type	The type of the table. Valid values: Physical Table and Logical Table.
Moved to Recycle Bin At	The date and time when the table is moved to the recycle bin, that is, the date and time when you delete the table in the governance console.
Processed At	The date and time when the table is processed in the recycle bin.
Status	The status of the table.

You can enter a keyword in the search box to search for tables whose names contain the keyword. You can also click the icon next to **All** and select a project to filter tables.

Clear a table

Tables cannot be restored after they are cleared from the recycle bin. Exercise caution when you perform this operation.

1. On the **Tables** page, find the table that you want to clear and click the icon in the **Actions** column.
2. In the **Clear Data** message, click **OK**.

Clear multiple tables

Tables cannot be restored after they are cleared from the recycle bin. Exercise caution when you perform this operation. You can clear multiple tables at a time.

1. On the **Tables** page, select multiple tables to be cleared and click **Clear** in the lower part of the page.
2. In the **Clear Data** message, click **OK**. If the status of the selected tables changes to

Succeeded, they are cleared from the recycle bin.

Restore a table

You can restore a table to the directory where the table was stored before it is moved to the recycle bin.

1. On the **Tables** page, find the table that you want to restore and click the icon in the **Actions** column.
2. In the **Restore Data** message, click **OK**.

Restore multiple tables

You can restore multiple tables to the directories where the tables were stored before they are moved to the recycle bin.

1. On the **Tables** page, select multiple tables that you want to restore and click **Restore** in the lower part of the page.
2. In the **Restore Data** message, click **OK**. If the status of the selected tables changes to **Succeeded**, they are restored from the recycle bin.

View the operational logs of a table

1. On the **Tables** page, find the table whose operational logs you want to view and click the icon in the **Actions** column.
2. In the **View Operations** message, view information about operations on the table, including the user who performed an operation, the time when the operation was performed, and the details of the operation.
3. Click **OK**.

9.14.5. Data quality

9.14.5.1. Overview

The Quality module of Dataphin provides a comprehensive data quality scheme that has various features. For example, you can configure quality rules to check data quality and use intelligent alerting.

Background information

As more demanding requirements for building, managing, and applying big data platforms emerge, Dataphin must adapt to more complex scenarios. The raw data from business systems may not meet specific requirements. To properly make business decisions based on data, you can configure quality rules in the Quality module of Dataphin to check the data quality, including the data timeliness, accuracy, integrity, consistency, and validity.

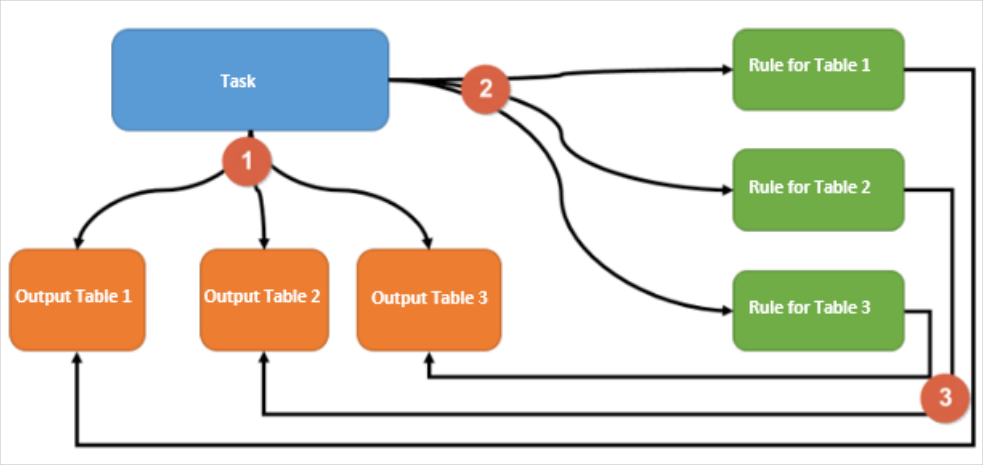
Features

You can use Quality to check the quality of various tables, trigger alerts, and generate quality reports.

The Quality module consists of the following submodules:

- **Quality Overview**: provides an overview of quality rule-based checks.
- **Quality Management**: allows you to configure quality rules. You can also view quality rules, check records, and quality reports.

The following figure shows the process of quality rule-based check.



9.14.5.2. Terms

This topic introduces the basic terms that are involved in the Quality module, such as the data object, quality rule, strong rule, and weak rule.

Term	Description
data object	An object that is managed in Quality. A data object can be a logical table or a physical table.
quality rule	A rule that is used to check the quality of a data object. A quality rule can check data quality at the granularity of fields. For example, you can configure a rule to monitor changes in the number of rows in a table at the ODS layer and check whether duplicate primary keys exist in the table.
strong rule	A rule whose check results affect the running of downstream nodes. If a data object fails a strong rule-based check, the system does not schedule downstream nodes of the current node where the data object resides. In addition, the system sets the status of the current node to failed and reports an alert to the specific alert recipients.
weak rule	A rule whose check results do not affect the running of downstream nodes. If a data object fails a weak rule-based check, the system does not forcibly stop scheduling downstream nodes of the current node, but only reports an alert to the specific alert recipients.
check record	A record that Quality generates after the quality of a data object is checked based on quality rules.
quality report	A report that Quality generates after the quality of a data object is checked based on quality rules. After the node where a data object resides is scheduled, Quality checks the quality of the updated data object based on the configured quality rules. Then, Quality generates a quality report that contains detailed results of the quality rule-based check.

Term	Description
partition expression	An expression that is used to define the data range to which a quality rule is applied.

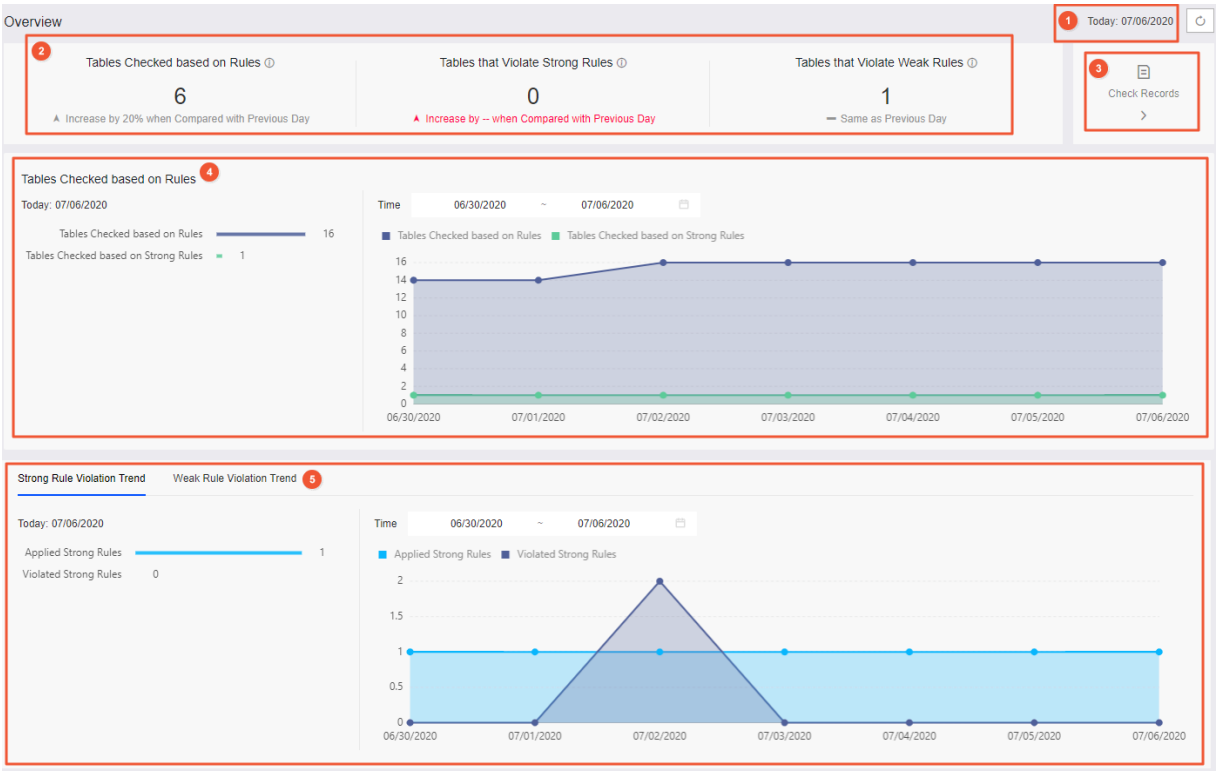
9.14.5.3. View statistics on the Overview page

The Overview page provides the overall results of quality rule-based checks on tables. It helps you properly identify and process abnormal check results.

To go to the **Overview** page, perform the following steps:

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.

On the **Overview** page, you can view the overall results of quality rule-based checks on tables.



No.	Description
1	Displays the date of the current day, which is used as the timestamp of data displayed on the Overview page.

No.	Description
2	• Displays the change in the number of tables that are checked based on quality rules on the current day, compared with that on the previous day. • Displays the change in the number of tables that fail strong rule-based checks on the current day, compared with that on the previous day. • Displays the change in the number of tables that fail weak rule-based checks on the current day, compared with that on the previous day.
3	Provides an entry for you to go to the Check Records page.
4	Displays the following data by default: • The number of tables that are checked based on quality rules on the current day and the trend within the latest seven days. • The number of tables that are checked based on strong rules on the current day and the trend within the latest seven days. Allows you to specify a time range within the last 30 days to view corresponding data.
5	• Displays the trend of statistics on strong rules, including: ◦ The number of strong rules that are used to check tables on the current day and the trend within the latest seven days. ◦ The number of strong rules that tables violate on the current day and the trend within the latest seven days. • Displays the trend of statistics on weak rules, including: ◦ The number of weak rules that are used to check tables on the current day and the trend within the latest seven days. ◦ The number of weak rules that tables violate on the current day and the trend within the latest seven days. Allows you to specify a time range within the last 30 days to view corresponding data.

9.14.5.4. Manage quality rules

9.14.5.4.1. View quality rules

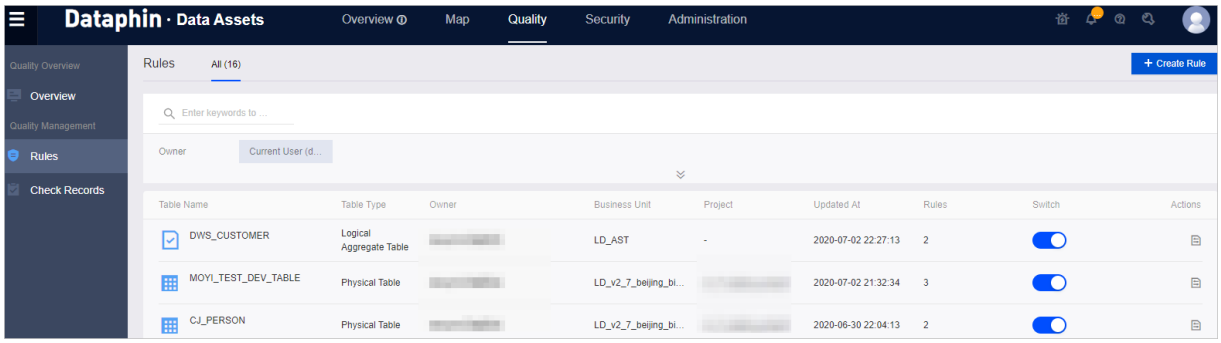
The Rules page displays information about quality rules that are configured for tables. This topic describes how to view quality rules and enable or disable quality rule-based check for tables.

Go to the Rules page

To go to the **Rules** page, perform the following steps:

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the Data Assets page, click **Quality** in the top navigation bar.
4. On the Quality page, click **Rules** in the left-side navigation pane.

On the Rules page, you can view information about quality rules that are configured for tables.



You can set **Owner**, **Business Unit**, **Project**, and **Table Type** to filter records. You can also enter a keyword in the search box to search for records of tables whose names contain the keyword.

Parameter	Description
Table Name	The name of the table for which quality rules are created.
Table Type	The type of the table. Valid values: • **Logical Aggregate Table** • **Logical Dimension Table** • **Logical Fact Table** • **Physical Table**
Owner	The owner of the table.
Business Unit	The business unit to which the table belongs. If the project to which a physical table belongs is not bound to a business unit, a hyphen (-) appears in the Business Unit column of the physical table.
Project	The project to which the table belongs.
Updated At	The last time when the quality rules of the table were updated.
Rules	The number of quality rules that are configured for the table.
Switch	Specifies whether quality rule-based check is enabled for the table. • indicates that quality rule-based check is enabled for the table. • indicates that quality rule-based check is disabled for the table.
Actions	The operations that you can perform on the table. You can click the icon in the **Actions** column to go to the **Rule Settings** page of the table.

View the settings of a rule

On the **Rules** page, find the table whose settings you want to view and click the icon in the **Actions** column. On the **Rule Settings** page, you can view the rule settings of the table.

Enable or disable quality rule-based check for a table

On the **Rules** page, find the table for which you want to disable or enable quality rule-based check and click the or icon in the **Switch** column.

9.14.5.4.2. Configure a quality rule

To properly manage data quality, you must configure quality rules. This topic describes how to configure a quality rule for a table.

Prerequisites

Tables are submitted or published to the production environment.

Step 1: Create a quality rule

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.
4. On the **Quality** page, click **Rules** in the left-side navigation pane.
5. On the **Rules** page, click **Create Rule** in the upper-right corner.
6. In the **Create Rule** dialog box, select a table and click **Configure**. You can select a table by using one of the following methods:
   - Click the icon and set **Business Unit**, **Project**, and **Table Type** to filter tables. Then, select the desired table from the drop-down list.
   - Click the arrow button. Enter a keyword in the search box to search for tables whose names contain the keyword and select the desired table.

Step 2: Configure alert information

After you configure alert information for a table, Alert Center reports alerts to the specified recipients if the table fails quality rule-based checks.

1. On the Rule Settings page of the table, click the icon in the **Alert Settings** section.
2. In the **Alert Settings** dialog box, set **Recipients** and **Methods** and click **OK**.

Step 3: Create a partition expression

A partition expression specifies the data range to which a group of quality rules are applied. You can create multiple partition expressions for a table and configure multiple quality rules under each partition expression, as shown in the following figure.

1. On the Rule Settings page of the table, click **Create Partition Expression** in the **Rule Settings** section.
2. In the **Create Partition Expression** dialog box, set **Type**, enter a partition expression, and then click **Save**.
   - If you set Type to a partition expression type that is built in Quality, the system automatically fills in the corresponding partition expression. The following table lists the partition expression types that are built in Quality and the corresponding partition

expressions.

Partition expression type	Partition expression
Last Day	ds='${yyyyMMdd}'
First Day of the Current Week	ds=tdBeginDate('${yyyyMMdd}', 'w', 'yyyyMMdd')
Last Day of the Current Week	ds=customEndDate('${yyyyMMdd}', 'w', '1', 0, 'yyyyMMdd')
First Day of the Current Month	ds=tdBeginDate('${yyyyMMdd}', 'm', 'yyyyMMdd')
Last Day of the Current Month	ds=customEndDate('${yyyyMMdd}', 'm', '01', 0, 'yyyyMMdd')
First Day of the Current Quarter	ds=tdBeginDate('${yyyyMMdd}', 'q', 'yyyyMMdd')
Last Day of the Current Quarter	ds=customEndDate('${yyyyMMdd}', 'q', '0101', 0, 'yyyyMMdd')
First Day of the Current Year	ds=tdBeginDate('${yyyyMMdd}', 'y', 'yyyyMMdd')
Last Day of the Current Year	ds=customEndDate('${yyyyMMdd}', 'y', '0101', 0, 'yyyyMMdd')
First Day of the Previous Month	ds=cBeginDate('${yyyyMMdd}', 'm', 'yyyyMMdd')
Last Day of the Previous Month	ds=cEndDate('${yyyyMMdd}', 'm', 'yyyyMMdd')
All Partitions	ds=ALL

- If you set Type to **Custom**, you must enter a partition expression, for example, `ds=${yyyyMM01}`.

After you create a partition expression, you can perform the following operations on it:

- Click the icon next to the partition expression to edit it.
- Click the icon next to the partition expression to delete it.

Step 4: Configure a quality rule

To configure a quality rule under a partition expression, perform the following steps:

1. On the Rule Settings page of the table, click **Create Rule** next to the partition expression.
2. In the **Create Rule** dialog box, set the parameters as prompted and click **Save**.The valid

values of the Rule Type parameter are **Template-based Rules** and **Custom Rules**.

- If you set Rule Type to **Template-based Rules**, you must set other parameters as described in the following table.

Parameter	Description
Object Name	The object to be checked based on the quality rule. The value of the **Object Name** parameter is in the `Table: Name of the current table` or `Field: Name of a field in the current table` format.
Strong or Weak Rule	The type of the quality rule. You can select Strong Rule or Weak Rule. ■ If you select **Strong Rule**, Alert Center reports an alert and the system blocks downstream nodes of the current node where the table resides when the specific object fails the rule-based check. ■ If you select **Weak Rule**, Alert Center reports an alert but the system does not block downstream nodes of the current node when the specific object fails the rule-based check.
Template	The check dimension of the quality rule. The valid values of the Template parameter vary based on the value you selected for the Object Name parameter. For more information, see Configuration table.
Trend	■ If the check type is fluctuation check, the valid values of the **Trend** parameter are **Absolute Value**, **Upward**, and **Downward**. ■ If the check type is comparison, the value of the **Trend** parameter is fixed to **Fixed Value**. For more information, see Configuration table.
Comparison Settings	The method for comparing the rule-based check result with the target value. You must specify a target value if the check type is comparison. The valid values of the **Comparison Method** parameter are **Greater Than Target Value**, **Greater Than or Equal To Target Value**, **Equal To Target Value**, **Less Than Target Value**, and **Less Than or Equal To Target Value**.
Fluctuation Threshold	The fluctuation threshold for the rule-based check result. The value of the **Threshold** parameter can range from 0.0 to 10.0.

Configuration table

Object name	Template	Trend	Configuration item	Check type
`Table: Name of the current table`	Compare the Target Table or Partition Size with that of Previous Day	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check

Object name	Template	Trend	Configuration item	Check type
Table: Name of the current table	Compare the Target Number of Table or Partition Rows with that of Previous Day	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Table or Partition Size with that of 7 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Number of Table or Partition Rows with that of 7 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Number of Table or Partition Rows with that of 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Number of Table or Partition Rows with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Average Number of Table or Partition Rows with that of 7 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Average Number of Table or Partition Rows with that of 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check

Object name	Template	Trend	Configuration item	Check type
Table: Name of the current table	Number of Table Partitions	Fixed Value	Comparison Settings	Comparison
Table: Name of the current table	Check Fluctuations in the Number of Table Partitions	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Table: Name of the current table	Compare the Target Number of Table or Partition Rows with that of Previous Day, the First Day of this Month, and that of 7, 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target and the Expected Number of Unique Values	Fixed Value	Comparison Settings	Comparison
Field: Name of a field in the current table	Compare the Target Number of Unique Field Values with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Average Field Value with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Maximum Field Value with that of Previous Day	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check

Object name	Template	Trend	Configuration item	Check type
Field: Name of a field in the current table	Compare the Target Maximum Field Value with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Minimum Field Value with that of Previous Day	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Minimum Field Value with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Sum of Field Values with that of Previous Day	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Compare the Target Sum of Field Values with that of Previous Day and that of 7 and 30 Days Ago	Absolute Value, Upward, and Downward	Fluctuation Threshold	Fluctuation check
Field: Name of a field in the current table	Number of Null Field Values	Fixed Value	Comparison Settings	Comparison
Field: Name of a field in the current table	Rate of Null Values (Number of Null Values/Total Number of Rows)	Fixed Value	Comparison Settings	Comparison

Object name	Template	Trend	Configuration item	Check type
Field: Name of a field in the current table	Number of Duplicate Field Values	Fixed Value	Comparison Settings	Comparison
Field: Name of a field in the current table	Rate of Duplicate Field Values (Number of Duplicate Values/Total Number of Rows)	Fixed Value	Comparison Settings	Comparison

- **If you set Rule Type to Custom Rules, you must set other parameters as described in the following table.**

Parameter	Description
Rule Name	The name of the quality rule.
Details	The content of the quality rule. Example: `select` `  sum(value) as metric` `from` `  current_table ctb` `  left outer join related_table rtb on ctb.id = rtb.id` `where` `  ds = ${bizdate};`
Strong or Weak Rule	The type of the quality rule. You can select Strong Rule or Weak Rule. ■ If you select **Strong Rule**, Alert Center reports an alert and the system blocks downstream nodes of the current node where the table resides when the specific object fails the rule-based check. ■ If you select **Weak Rule**, Alert Center reports an alert but the system does not block downstream nodes of the current node when the specific object fails the rule-based check.
Check Type	The check type of the quality rule. The valid values of the **Check Type** parameter are **Compare Target Data with that of Previous Day**, **Compare Target Data with that of 7 Days Ago**, **Compare the Target Average Value with that of 7 Days Ago**, **Compare the Target Average Value with that of 30 Days Ago**, and **Compare with Fixed Value**.

Parameter	Description
Trend	■ If the check type is fluctuation check, the valid values of the **Trend** parameter are **Absolute Value**, **Upward**, and **Downward**. ■ If the check type is comparison, the value of the **Trend** parameter is fixed to **Fixed Value**.
Comparison Settings	The method for comparing the rule-based check result with the target value. You must specify a target value if the check type is comparison. The valid values of the **Comparison Method** parameter are **Greater Than Target Value**, **Greater Than or Equal To Target Value**, **Equal To Target Value**, **Less Than Target Value**, and **Less Than or Equal To Target Value**.
Fluctuation Threshold	The fluctuation threshold for the rule-based check result. The value of the **Threshold** parameter can range from 0.0 to 10.0.

3. Select the configured quality rule in the rule list and click **Test Run** in the lower part of the list. After the test is successful, turn on the switch for the quality rule. Then, the quality rule takes effect.The test run can in one of the following states:
   - **Pending**
   - **Succeeded**
   - **Failed**
   - **Running**

9.14.5.4.3. Modify a quality rule

To properly manage data quality, you must configure quality rules. This topic describes how to modify a quality rule for a table.

Prerequisites

A quality rule is configured for a table. For more information, see Configure a quality rule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.
4. On the **Quality** page, click **Rules** in the left-side navigation pane.
5. On the **Rules** page, find the table for which you want to modify a quality rule and click the icon in the **Actions** column.
6. On the **Rule Settings** page, find the quality rule that you want to modify and click the icon in the **Actions** column.
7. In the **Change Rule** dialog box, modify the parameter settings as prompted. For more information, see Configure a quality rule.

8. Click **Save**.

9.14.5.4.4. Delete a quality rule

To properly manage data quality, you must configure quality rules. This topic describes how to delete a quality rule for a table.

Prerequisites

A quality rule is configured for a table. For more information, see Configure a quality rule.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.
4. On the **Quality** page, click **Rules** in the left-side navigation pane.
5. On the **Rules** page, find the table for which you want to delete a quality rule and click the icon in the **Actions** column.
6. On the **Rule Settings** page, find the quality rule that you want to delete and click the icon in the **Actions** column.
7. In the **Delete Rule** message, click **OK**.

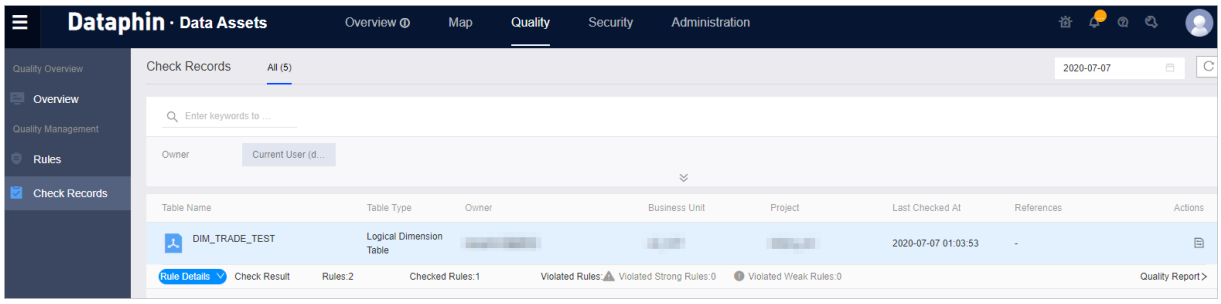
9.14.5.5. View check records

The Check Records page displays records of quality rule-based checks on tables. This topic describes how to view such records.

To go to the **Check Records** page, perform the following steps:

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Data Assets** in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.
4. On the **Quality** page, click **Check Records** in the left-side navigation pane.

On the Check Records page, you can click the icon in the upper-right corner to select a date. Then, check records of that day appear on the page.



You can set **Owner**, **Business Unit**, **Project**, **Status**, **Table Type**, and **Rule Template** to filter check records. You can also enter a keyword in the search box to search for check records of tables whose names contain the keyword.

GUI element	Description
Table Name	The name of the table that Quality checked based on quality rules.
Table Type	The type of the table.
Owner	The owner of the table.
Business Unit	The business unit to which the table belongs.
Project	The project to which the table belongs.
Last Checked At	The last time when the table was checked based on quality rules.
References	The number of times that the table was referenced by tasks when the tasks were scheduled in the production environment.
Actions	The operations that you can perform on the table. You can click the icon in the Actions column to go to the Rule Settings page of the table, where you can: • View existing quality rules of the table. • Create quality rules for the table. For more information, see Configure a quality rule. • Modify existing quality rules of the table. For more information, see Configure a quality rule.
Rule Details	The details about quality rule-based checks on the table. You can click the icon next to **Rule Details** to view the details.

GUI element	Description
Check Result	The summary of quality rule-based checks on the table, including: • **Rules**: the number of quality rules that have been configured for the table. • **Checked Rules**: the number of quality rules that Quality used to check the table. • **Violated Rules**: ○ **Violated Strong Rules**: the number of strong rules that the table violated. ○ **Violated Weak Rules**: the number of weak rules that the table violated.
Quality Report	The quality report of the table. You can click **Quality Report** to go to the Quality Report page of the table. You can view information such as the quality analysis result, report details, and rule-based check instances on the page. For more information, see View a quality report.

9.14.5.6. View a quality report

After a task that references a table is scheduled in the production environment, Quality checks the quality of the updated table based on the quality rules configured for the table. Then, Quality generates a quality report that contains detailed results of the quality rule-based check. This topic describes how to view a quality report.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click Data Assets in the top navigation bar.
3. On the **Data Assets** page, click **Quality** in the top navigation bar.
4. The **Quality** page appears.
5. Go to the **Quality Report** page of a table by using one of the following methods:
   - a. On the Quality page, click **Rules** in the left-side navigation pane.
     b. On the **Rules** page, find the table whose quality report you want to view and click the icon in the **Actions** column.
     c. On the **Rule Settings** page of the table, click **Quality Report**.
   - a. On the Quality page, click **Check Records** in the left-side navigation pane.
     b. On the **Check Records** page, find the table whose quality report you want to view and click **Quality Report** under the table.
6. On the **Quality Report** page of the table, click the icon in the upper-right corner to select a date. Quality displays the quality report of the table on that day.

Parameter or section	Section	Parameter	Description
Table Details	N/A	Table	The name of the table.
		Business Unit	The business unit to which the table belongs.
		Project	The project to which the table belongs.
		Table Type	The type of the table. Valid values: ○ Logical Aggregate Table ○ Logical Dimension Table ○ Logical Fact Table ○ Physical Table
		Owner	The owner of the table.
		Task Node	The task that referenced the table in the production environment.
		Alert Recipients	The recipients to whom Alert Center reported alerts when the table failed quality rule-based checks.
Quality Analysis Report	N/A	Violated Rules	The number of quality rules that the table violated.
		Checked Rules	The total number of quality rules that are configured for the table and the number of quality rules that Quality used to check the table.
		Instance Execution Time	The time when the task that referenced the table was run in the production environment.
	N/A	Partition Expression	The partition expression of the table.
		Partition	The partition key value that is obtained based on the partition expression, for example, `DS=20200212`.
		Field Name	The name of the field for which a quality rule is configured under the partition expression.
		Rule Template Name	The template of the quality rule that is configured for the field.
		Strong or Weak Rule	The type of the quality rule.

Parameter or section	Section	Parameter	Description
Report Details	Field Check	Comparison Method	The method for comparing the quality rule-based check result with the specific fluctuation threshold or target value. ○ If the check type is fluctuation check, the valid values of the **Trend** parameter are **Absolute Value**, **Upward**, and **Downward**. ○ If the check type is comparison, the value of the **Trend** parameter is fixed to **Fixed Value**.
		Threshold	The fluctuation threshold of the field values.
		Status	The result of the quality rule-based check on the field.
		Details	The details about the field values. To view the value trend of the field, perform the following steps: i. Click the icon in the **Details** column of the field. ii. In the **Alert Trend** dialog box, specify a time range in the upper-right corner and view the value trend that appears. iii. Click **OK**.
	Custom Rule	Custom Rule Name	The name of the custom quality rule.
		Strong or Weak Rule	The type of the custom quality rule.
		Comparison Method	The method for comparing the custom quality rule-based check result with the specific fluctuation threshold or target value. ○ If the check type is fluctuation check, the valid values of the **Trend** parameter are **Absolute Value**, **Upward**, and **Downward**. ○ If the check type is comparison, the value of the **Trend** parameter is fixed to **Fixed Value**.
		Status	The result of the custom quality rule-based check on the field.

Parameter or section	Section	Parameter	Description
		Details	The details about the field values. To view the value trend of the field, perform the following steps: i. Click the icon in the **Details** column of the field. ii. In the **Alert Trend** dialog box, specify a time range in the upper-right corner and view the value trend that appears. iii. Click **OK**.
	Rule-based Check Instances	N/A	**Table Name**
Partition			The partition that Quality checked based on the quality rule.
Rule			The quality rule that Quality used to check the partition.
Owner			The owner of the table.
Date Timestamp			The data timestamp that Quality specified based on the partition.
Execution Time (Range)			The duration of the quality rule-based check.
Status			The result of the quality rule-based check. Valid values: ○ **Failed** ○ **Running** ○ **Succeeded**
Log			The operational logs of the quality rule-based check instance. You can click the icon in the **Log** column to view the operational logs of the quality rule-based check instance.

9.15. Theme-based data service

9.15.1. Ad hoc query

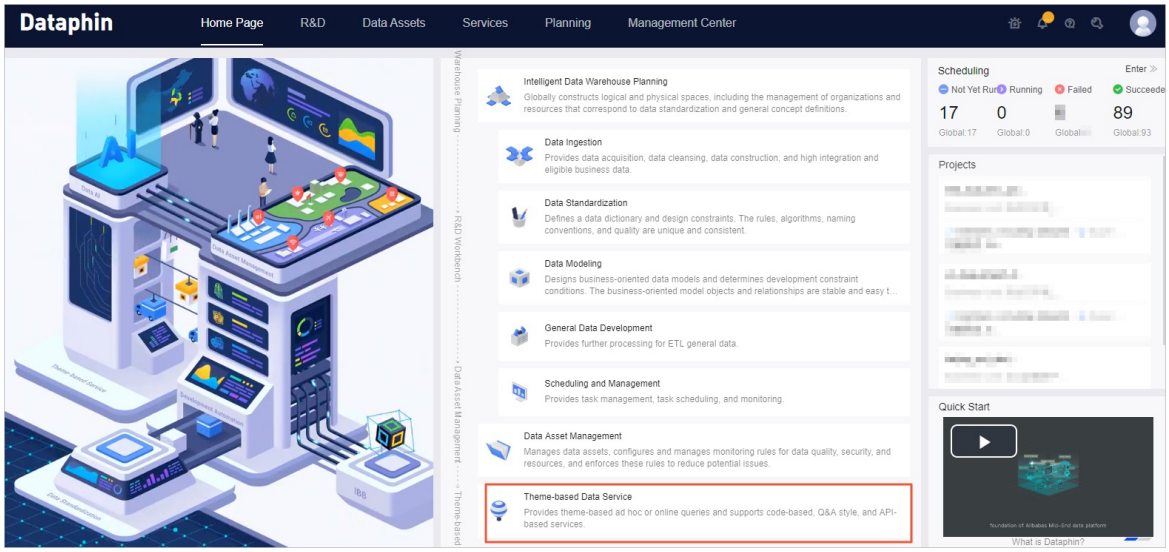
Dataphin provides the ad hoc query feature for you to query data.

Context

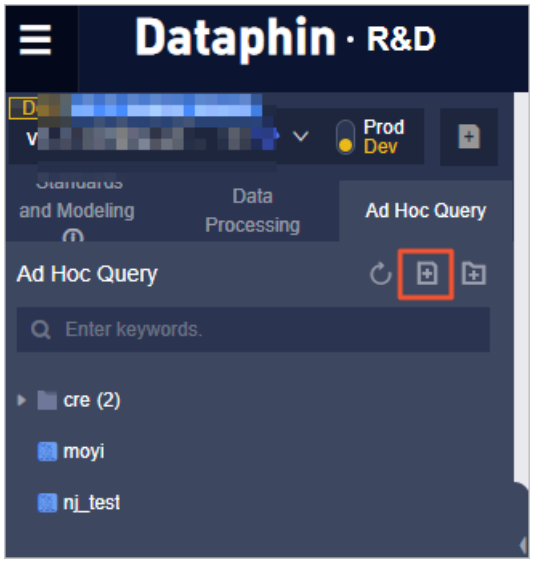
If your computing engine type is Hadoop, the Hive SQL syntax is supported in ad hoc queries. If your computing engine type is MaxCompute, the MaxCompute SQL syntax is supported in ad hoc queries. Dataphin can automatically identify and change the SQL syntax based on your computing engine type.

Procedure

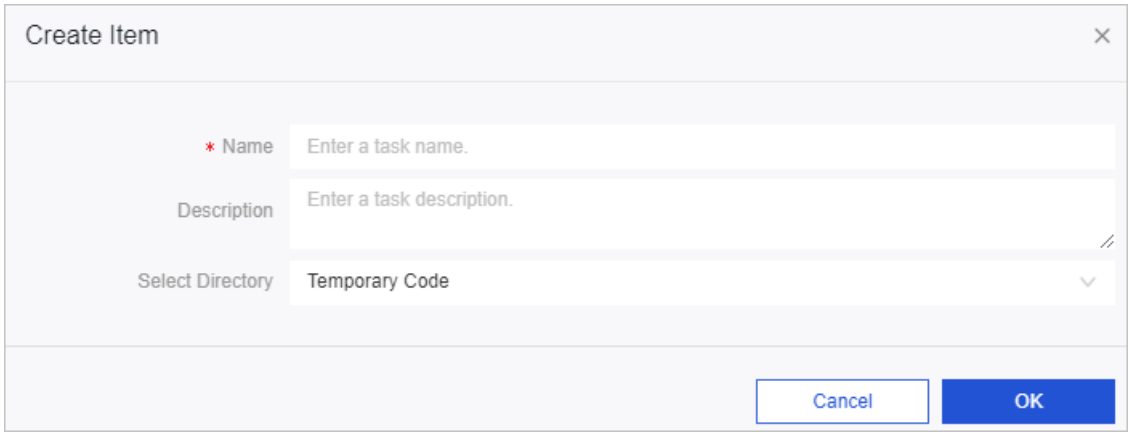
1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Theme-based Data Service** to go to the Ad Hoc Query tab.



3. On the Ad Hoc Query tab of the Develop tab, click the **Create Item** icon in the left-side navigation pane.



4. In the **Create Item** dialog box that appears, enter the task name and description, select a directory, and then click **OK**.



5. In the left-side ad hoc query task list, click the ad hoc query task that you created in the preceding step to go to the **Code Editor** tab.
6. Write SQL statements, click the **Save** icon, and then click **Run** in the upper-right corner.

> **Note** When you write SQL statements, reference table names in the format of *Project name.Table name*, for example, *demo.dim_qwe*.

7. After SQL statements are run, view the results on the **Result** tab.

9.16. Data services

9.16.1. Overview

As the last step in constructing a Data Mid-End based on Dataphin, the Services module provides data services in a uniform manner. In the Services module, Dataphin centrally manages data services. This effectively facilitates and secures the use of data services.

Data services API Service Workbench Administration
Dataphin is designed to build a uniform data service bus to help enterprises increase the value of their data assets while guaranteeing data reliability, security, and effectiveness. The Services module allows you to create API operations in the easy-to-use template wizard mode. This helps enterprises maximize the value of data applications that they create.

The Services module of Dataphin consists of the **API Service**, **Workbench**, and **Administration** pages.

- The **API Service** page lists all published API operations.
- The **Workbench** page is composed of **Development Center** and **Application Center**.
  - **Development Center** is designed for data service providers to develop services. It contains two modules: **Service Units** and **APIs**.

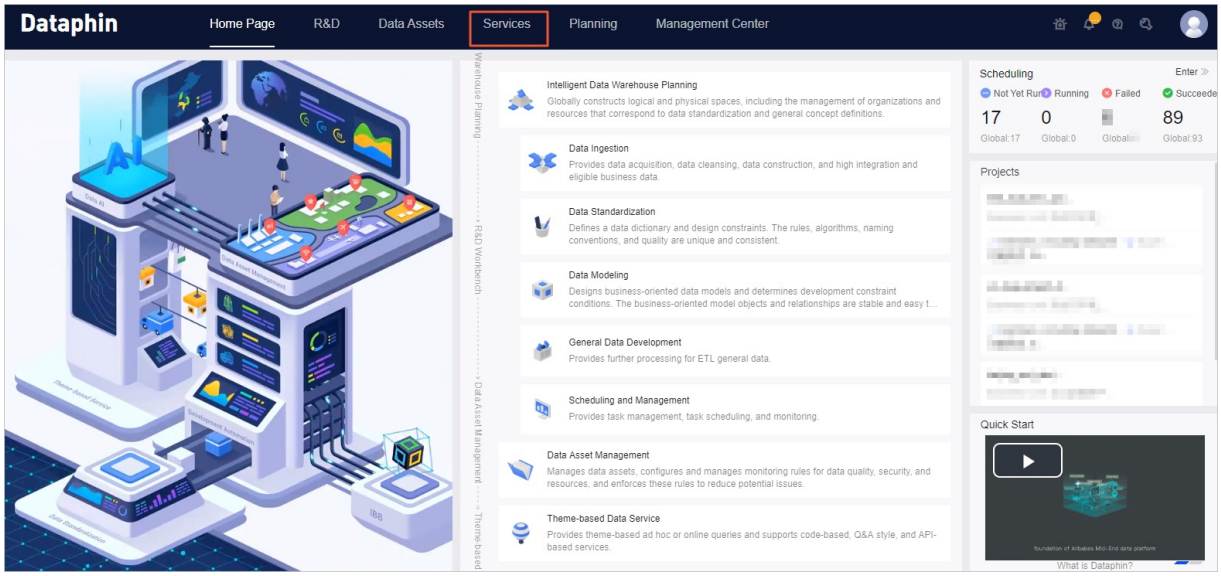
  - **Application Center** is designed for data service users to consume API services from applications. It contains two modules: **APIs** and **Applications**.

- The **Administration** page contains the following modules: **Member Management**, **Groups**, **Metadata Management**, **Monitoring**, **Network Configuration**, Call Examples, and Dataphin Data Sources. On this page, you can monitor and manage data services in a uniform manner.

In the Services module, you can develop, call, and monitor API operations.

Feature	Procedure
Develop API operations	1. Add a Dataphin user to the Services module as a **developer**. For more information, see Manage members. Note Only a developer can create API operations in the Services module. 2. Create a group. For more information, see Manage service unit groups. 3. Create a service unit. For more information, see Create a service unit. 4. Create an API operation. For more information, see Create an API operation. 5. Test the created API operation. For more information, see Test an API operation. 6. Publish the API operation. For more information, see Publish an API operation.
Call API operations	1. Add a Dataphin user to the Services module as an **application user**. For more information, see Manage members. Note Only an application user can call API operations in the Services module. 2. Create an application. For more information, see Create an application. 3. Apply for the permission to use an API operation. For more information, see API Service. 4. Debug the requested API operation. For more information, see Debug an API operation. 5. Call the API operation. For more information, see Manage call examples.
Monitor API operations	For more information about how to monitor API operations, see Monitor API operations.

Log on to the Dataphin console. On the Dataphin homepage, click Services in the top navigation bar. The Service page appears.



9.16.2. Terms

This topic describes the terms of the Services module in Dataphin, such as Dataphin data source, API operation, group, service unit, and application.

Term	Description
Dataphin data source	The data source that is created based on a table.
API operation	The API operation that is created by configuring a table.
Service unit	Allows you to manage the metadata of one or more tables in a uniform manner.
Application	The permission subject that allows members to consume data services, including calling API operations and using Dataphin data sources.
Group	Allows you to classify and manage data services and applications based on specific features or scenarios.
API Service page	The page that displays the API operations that you have published. It is the API market of Dataphin.
Metadata management	Allows you to configure and manage the schema mapping from NoSQL databases to relational databases to guarantee the consistency of data sources.
Network configuration	The network configuration for an application to access API operations and data sources. Network configuration includes the domain name configuration and the configuration of the Virtual Private Cloud (VPC) whitelist.

Term	Description
Call Examples page	The page that provides the instructions for calling API operations and using Dataphin data sources.

9.16.3. Configure the network

Before an application can call API operations and use Dataphin data sources, you must configure the network. Network configuration includes the domain name configuration and the configuration of the VPC whitelist. This topic describes how to enable and disable a second-level domain name, configure an independent domain name, and unbind an independent domain name to the current Apsara Stack tenant account.

Context

You can configure the network only by using an Apsara Stack tenant account.

Enable and disable a second-level domain name

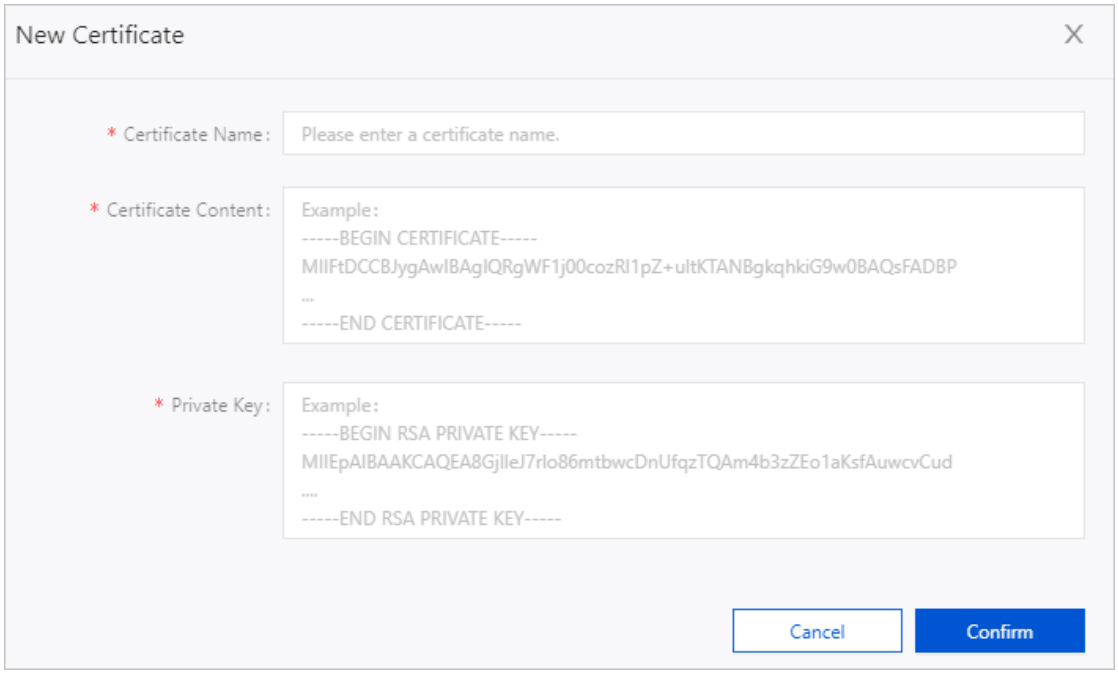
This section describes how to enable and disable the second-level domain name on the Internet. You can follow the same procedure to enable and disable the second-level domain name in a VPC.

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Services** in the top navigation bar.
3. On the Service page, click **Administration** in the top navigation bar.
4. On the Administration page, click **Network Configuration** in the left-side navigation pane.
5. Disable the second-level domain name.
   i. Click **Close the public domain second-level domain name** next to the domain name.
   ii. In the **Prompt** message, click **Confirm**.
6. Enable the second-level domain name.
   i. Click **Open the second-level domain name of the public network**.
   ii. In the **Prompt** message, click **Confirm**.

Configure an independent domain name

1. Bind an independent domain name to the current Apsara Stack tenant account.
   i. On the **Domain Configuration** tab, click **Bind Domain Name** in the upper-right corner of the **Independent Domain Name** section.
   ii. In the **Bind Domain Name** dialog box, enter a domain name.
   iii. Click **Confirm** to bind the independent domain name to the current Apsara Stack tenant account.
2. Bind an SSL certificate to the independent domain name.
   i. Click **Add** in the **SSL Certificate** column.

ii. In the **New Certificate** dialog box, enter a certificate name, the certificate content, and the private key as required.



Parameter	Description
Certificate Name	The name of the SSL certificate.
Certificate Content	The content of the SSL certificate. Enter the content based on the example on the page.
Private Key	The private key of the SSL certificate. Enter the private key based on the example on the page.

iii. Click **Confirm** to bind the SSL certificate.

Unbind an independent domain name

1. In the Independent Domain Name section, find the independent domain name that you want to unbind and click the icon in the Actions column.
2. In the message that appears, click **OK**.

9.16.4. Add a member

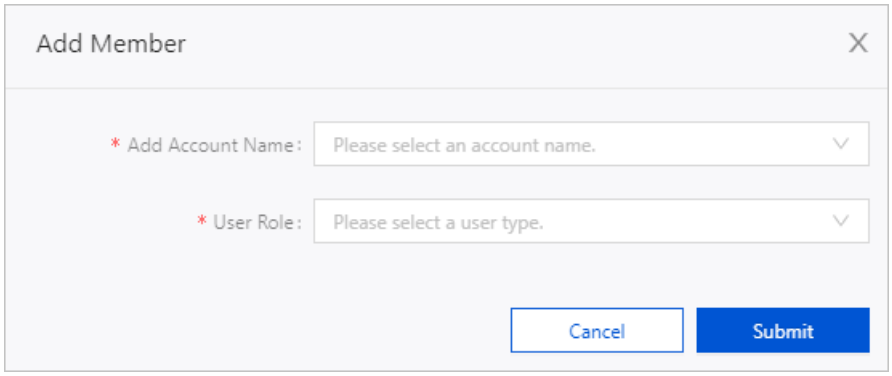
The Services module allows you to centrally manage the members who develop and use data services in Dataphin. This topic describes how to add a member to the Services module.

Context

A Dataphin user can access the Services module only after the user is added as a member with the **Developer** or **Application User** role. Only members can develop and use API operations and data sources.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Services** in the top navigation bar.
3. On the Service page, click **Administration** in the top navigation bar.
4. On the Administration page, click **Member Management** in the left-side navigation pane.
5. On the **Member Management** page, click **Add Member** in the upper-right corner.
6. In the **Add Member** dialog box, select the name of the account to be added as a member and the user type from the **Add Account Name** and **User Role** drop-down lists, respectively.



Parameter	Description
Add Account Name	The name of the account to be added as a member of the Services module.
User Role	The role to be assigned to each selected account. Valid values: ○ **Developer**: Developers provide data services and develop API operations. ○ **Application User**: Application users use data services and call API operations.

7. Click **Submit**.

The following table describes the operations that you can perform on the added members.

Operation	Description
Change the user role of a member	Click the icon in the **Actions** column to change the user role of a member.
Delete a member	○ Click the icon in the **Actions** column to delete a member from the Services module. ○ To delete multiple members at a time, select the members and click **Remove** in the upper-right corner of the **Member Management** page.

9.16.5. Develop API operations

9.16.5.1. Create a group

Dataphin allows you to manage service units and applications by group. This topic describes how to create a group for managing service units and applications.

Procedure

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Services** in the top navigation bar.
3. On the Service page, click **Administration** in the top navigation bar.
4. On the Administration page, click **Groups** in the left-side navigation pane.
5. On the **Service Unit** tab, click **Create Group** in the lower part of the group list.

6. Enter a name in the field that appears and click **OK**. A service unit group is created.
7. Click the **Application** tab. On the **Application** tab, click **Create Group** in the lower part of the group list. Enter a name in the field that appears and click **OK**. An application group is created.

   The following table describes the operations that you can perform on the created groups.

Operation	Description
Change the name of a group	Click the icon in the **Actions** column to change the name of a group.
Delete a group	Click the icon in the **Actions** column to delete a group.

9.16.5.2. Create a service unit

9.16.5.2.1. Create a metadata set

By using metadata, you can configure and manage the schema mapping from NoSQL databases to relational databases to ensure the consistency of data sources. This topic describes how to create metadata sets for HBase, Elasticsearch, and MongoDB data sources.

Context

Before you create a service unit based on an HBase, an Elasticsearch, or a MongoDB data source, you must create a metadata set for the corresponding data source.

Create a metadata set for an HBase data source

1. Log on to the Dataphin console.
2. On the Dataphin homepage, click **Services** in the top navigation bar.
3. On the Service page, click **Administration** in the top navigation bar.
4. On the Administration page, click **Metadata Management** in the left-side navigation pane.
5. On the **Metadata Management** page, click **Create Metadata** in the upper-right corner.
6. On the **Create Metadata Set** page, specify a source table and add rowkeys and columns as required.

Section	GUI element	Description
Data Table Configuration	Physical Data Table	The configuration that specifies the source table. Select HBASE, a data source, and a physical table from the drop-down lists in sequence.
Rowkey Configuration	Delimiters	The delimiter used to separate different rowkeys.
	Rowkey Field Name	The name of the rowkey.
	Data Type	The data type of the rowkey.
	Row Key Description	The description of the rowkey.
	Actions	The operations that you can perform on the rowkey. You can click the icon in the **Actions** column to delete a field.
Column Field Configuration	Column Family Name	The name of the column family to which the column belongs.
	Column Qualifier Name	The name of the column.
	Data Type	The data type of the column.
	Description	The description of the column.
	Actions	The operations that you can perform on the column. You can click the icon in the **Actions** column to delete a column.

7. Click **Submit**.

Create a metadata set for an Elasticsearch data source

1. On the **Metadata Management** page, click **Create Metadata** in the upper-right corner.
2. On the **Create Metadata Set** page, specify a source table and add fields as required.

Section	GUI element	Description
Data Table Configuration	Physical Data Table	The configuration that specifies the source table. Select ES, a data source, and a physical table from the drop-down lists in sequence.
Field Configuration	Field Name	The name of the field. To configure a nested field, separate multiple field names with periods (.).
	Data Type	The data type of the field.
	Description	The description of the field.
	Actions	The operations that you can perform on the field. You can click the icon in the **Actions** column to delete a field.

3. Click **Submit**.

Create a metadata set for a MongoDB data source

The configurations for creating a metadata set for a MongoDB data source are similar to those for an Elasticsearch data source. For more information, see Create a metadata set for an Elasticsearch data source.

9.16.5.2.2. Create a service unit from a physical table

Dataphin allows you to create a service unit to manage the metadata of a physical table. This topic describes how to create a service unit from a physical table.

Prerequisites

- A service unit group is created. For more information, see Create a group.
- The corresponding metadata set is created if the source physical table resides in an HBase, an Elasticsearch, or a MongoDB data source. For more information, see Create a metadata set.

Context

The fields of a service unit can be referenced by multiple API operations.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Service Units** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. In the left-side navigation pane, click **Service Units**.
3. On the **Service Units** page, click **Create Service Unit** in the upper-right corner. The Create Service Unit page appears.
4. Select a service unit type.
   i. In the Service Unit Type Selection step, select **Single Physical Table Service Unit**

i. In the Service Unit Type Selection step, select **Single Physical Table Service Unit**.

ii. Click **Next**.

5. Configure the service unit.

i. In the **Service Unit Basic Information Configuration** step, set the parameters as required.

Parameter	Description
Service unit name	The name of the service unit.
Service Unit Group	The group to which the service unit belongs.
Physical Data Table	The source table for the service unit. To specify a source table, perform the following steps: a. Select a data source type. b. Select a data source. c. Select a table.
Service unit description	The description of the service unit.

ii. Click **Next**.

6. Configure the fields of the service unit.

i. In the **Service Unit Field Configuration** step, configure the fields of the service unit as required.

ii. Click **Data Preview** to preview the data of the service unit to create.

iii. Click **Identify and Publish in the Staging Environment** to create and publish the service unit in the staging environment.

7. On the **Service Units** page, find the created service unit, move the pointer over the icon in the Actions column, and then select **Publish** to publish the service unit to the production environment.

The following table describes the operations that you can perform on the created service units.

Operation	Description
Modify the configuration of a service unit	Click the icon in the **Actions** column to modify the configuration of a service unit.

Operation	Description
Delete a service unit	Move the pointer over the More icon in the **Actions** column and select the icon to delete a service unit. **Note** ○ You can delete only unpublished service units. ○ To delete a service unit, you must be the owner of the service unit or the super administrator.
Unpublish a service unit	Move the pointer over the icon in the **Actions** column and select **Unpublish** to unpublish a service unit that has been previously published to the production environment.
Manage the members of a service unit	Click the icon in the **Actions** column. The Service Unit Member Management page of the service unit appears. ○ Click **Add Member**. In the Add Members to Service Unit dialog box, select the name of the account to be added as a member and the user type from the **Select Member** and **Select Role** drop-down lists, respectively. Then, click **Submit** to add a member to the service unit. ○ Click the icon in the **Actions** column and select a member from the Transfer Ownership To drop-down list to transfer the ownership of the service unit to the selected member. ○ Click the icon in the **Actions** column to delete a member from the service unit. ○ To delete multiple members at a time, select the members and click **Delete Members** in the upper-right corner.

9.16.5.2.3. Create a service unit from multiple physical tables

Dataphin allows you to create a service unit to manage the metadata of multiple physical tables. This topic describes how to create a service unit from multiple homogeneous or heterogeneous physical tables.

Prerequisites

- A service unit group is created. For more information, see Create a group.
- The corresponding metadata sets are created if the source physical tables reside in HBase, Elasticsearch, or MongoDB data sources. For more information, see Create a metadata set.

Context

The fields of a service unit can be referenced by multiple API operations.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Service Units** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. In the left-side navigation pane, click **Service Units**.
3. On the **Service Units** page, click **Create Service Unit** in the upper-right corner. The Create Service Unit page appears.
4. Select a service unit type.
   i. In the Service Unit Type Selection step, select **Multiple Physical Table Service Unit**.
   ii. Click **Next**.
5. Configure the service unit.

i. In the **Service Unit Basic Information Configuration** step, set the parameters as required.

GUI element	Description
Service unit name	The name of the service unit.
Service Unit Group	The group to which the service unit belongs.
Service unit description	The description of the service unit.
Add Physical Table	To add a physical table, perform the following steps: a. In the Mounted Physical Table List section, click **Add Physical Table**. b. In the **Add Physical Table** dialog box, select a data source type, a data source, and a table in sequence to specify a source table. If the required data source is unavailable, click **Create Data Source** to go to the Planning page and create a data source. For more information, see Create a MaxCompute data source. c. In the **No field is selected** section, select the fields of the specified physical table as required and click the > icon to add the fields to the **Selected Field** section. To remove a field from the **Selected Field** section, select the field and click the < icon. d. Click **OK**.

By default, the system sets the first physical table that you add as the primary table. You can click the icon in the **Primary Table** column to change the primary table. You can also perform the following operations on the added physical tables:

- Click the icon in the **Actions** column to modify the configuration of a physical table.
- Click the icon in the **Actions** column to delete a physical table.

ii. Click **Next**.

6. Configure the fields of the service unit.

i. In the **Service Unit Field Configuration** step, configure the fields of the service unit as required.

Section	Parameter	Description
Service Unit Association Field Configuration	Service Unit Field	The name of the field in the service unit.
	Physical table name. Associated Field	The name of the association field. An association field is used to associate the physical tables that you added.
	Service Unit Association Field Type	The data type of the association field.
	Actions	The operations that you can perform on the association field. You can click the icon in the **Actions** column to delete a configured association field. **Note** You cannot delete the association field in the first line.
Service Unit Field Configuration	Service Unit Field Type	The data type of the field of the service unit. Dataphin automatically selects the data type of each field based on the data type of the field in the source table.
	Service Unit Field	The name of the field. Dataphin automatically generates the name for each field based on the name of the field in the source table.

ii. Click **Data Preview** to preview the data of the service unit to create.

iii. Click **Identify and Publish in the Staging Environment** to create and publish the service unit in the staging environment.

7. On the **Service Units** page, find the created service unit, move the pointer over the icon in the Actions column, and then select **Publish** to publish the service unit to the production environment.

   The following table describes the operations that you can perform on the created service units.

Operation	Description
Modify the configuration of a service unit	Click the icon in the **Actions** column to modify the configuration of a service unit.

Operation	Description
Delete a service unit	Move the pointer over the More icon in the **Actions** column and click the icon to delete a service unit. **Note** ◦ You can delete only unpublished service units. ◦ To delete a service unit, you must be the owner of the service unit or the super administrator.
Unpublish a service unit	Move the pointer over the icon in the **Actions** column and select **Unpublish** to unpublish a service unit that has been previously published to the production environment.
Manage the members of a service unit	Click the icon in the **Actions** column. The Service Unit Member Management page of the service unit appears. ◦ Click **Add Member**. In the Add Members to Service Unit dialog box, select the name of the account to be added as a member and the user type from the **Select Member** and **Select Role** drop-down lists, respectively. Then, click **Submit** to add a member to the service unit. ◦ Click the icon in the **Actions** column and select a member from the Transfer Ownership To drop-down list to transfer the ownership of the service unit to the selected member. ◦ Click the icon in the **Actions** column to delete a member from the service unit. ◦ To delete multiple members at a time, select the members and click **Delete Members** in the upper-right corner.

9.16.5.3. Create an API operation

9.16.5.3.1. Create an API operation in template wizard mode

Dataphin allows you to create an API operation by specifying a service unit and configuring the mapping between the fields of the service unit and the request and response parameters of the API operation on the GUI. This topic describes how to create an API operation in template wizard mode.

Prerequisites

- A service unit group is created. For more information, see Create a group.

- A service unit is created. For more information, see Create a service unit from multiple physical tables and Create a service unit from a physical table.

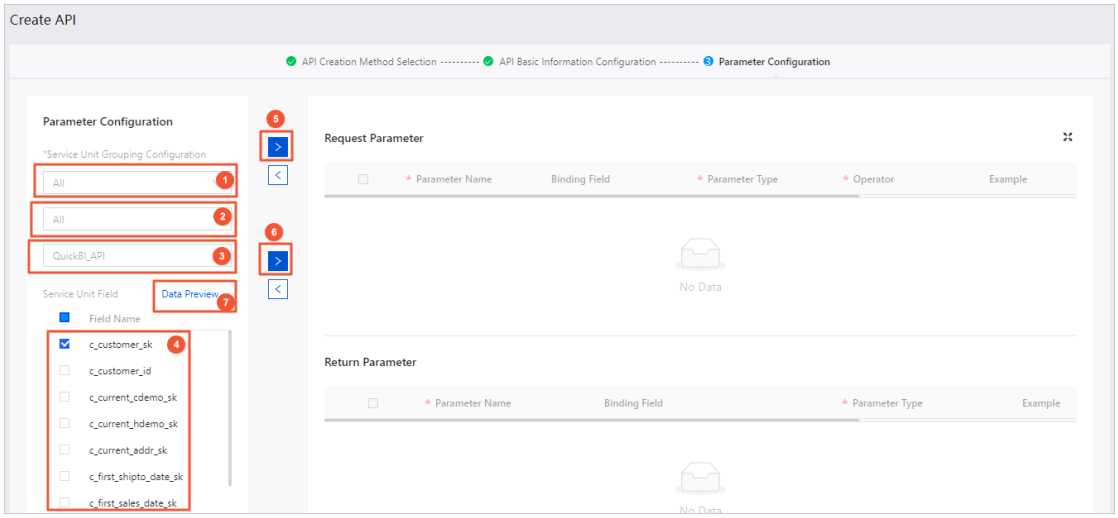
Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.
3. On the APIs page, click **Create API** in the upper-right corner.
4. Select a creation method.
   i. In the **API Creation Method Selection** step, select **Template Wizard Mode**.
   ii. Click **Next**.
5. Configure the API operation.
   i. In the **API Basic Information Configuration** step, set the parameters as required

Parameter	Description
API Name	The name of the API operation. The name can contain letters, digits, and underscores (_).
Request Method	The method for calling the API operation. Valid values: ■ GET ■ LIST
Data Update Frequency	The frequency at which data is synchronized by using the API operation. Valid values: ■ Daily ■ Hourly ■ Minutely
Description	The description of the API operation.
Protocol	The protocol that is used by the API operation to synchronize data. Valid values: ■ HTTP ■ HTTPS

   ii. Click **Next Step**.
6. Configure the request and response parameters for the API operation.

i. In the **Parameter Configuration** step, set the parameters as required.



No.	GUI element	Description
1	Service Unit Grouping Configuration	Click the icon and select a service unit group.
2		Click the icon and select a service unit type.
3		Click the icon and select a service unit.
4	Service Unit Field	Select fields of the service unit.
5	N/A	After you complete the preceding configuration in sections marked as 1 to 4, click the icon marked as 5 to add the selected fields to the **Request Parameter** section. Then, set **Parameter Name**, **Parameter Type**, **Operator**, **Example**, **Description**, and **Required** for the added request parameters.
6	N/A	After you complete the preceding configuration in sections marked as 1 to 4, click the icon marked as 6 to add the selected fields to the **Return Parameter** section. Then, set **Parameter Name**, **Parameter Type**, **Example**, and **Description** for the added response parameters.
7	Data Preview	Click **Data Preview** to preview the data of the selected service unit.

ii. Click **Identify and Publish in the Staging Environment** to create and publish the API operation in the staging environment.

9.16.5.3.2. Create an API operation in custom SQL mode

Dataphin allows you to create an API operation by specifying a service unit and writing an SQL statement. Dataphin parses the request and response parameters of the API operation based on the SQL statement. This topic describes how to create an API operation in custom SQL mode.

Prerequisites

- A service unit group is created. For more information, see Create a group.
- A service unit is created. For more information, see Create a service unit from multiple physical tables and Create a service unit from a physical table.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.
3. On the APIs page, click **Create API** in the upper-right corner.
4. Select a creation method.
   i. In the **API Creation Method Selection** step, select **Custom SQL Mode**.
   ii. Click **Next**.
5. Configure the API operation.

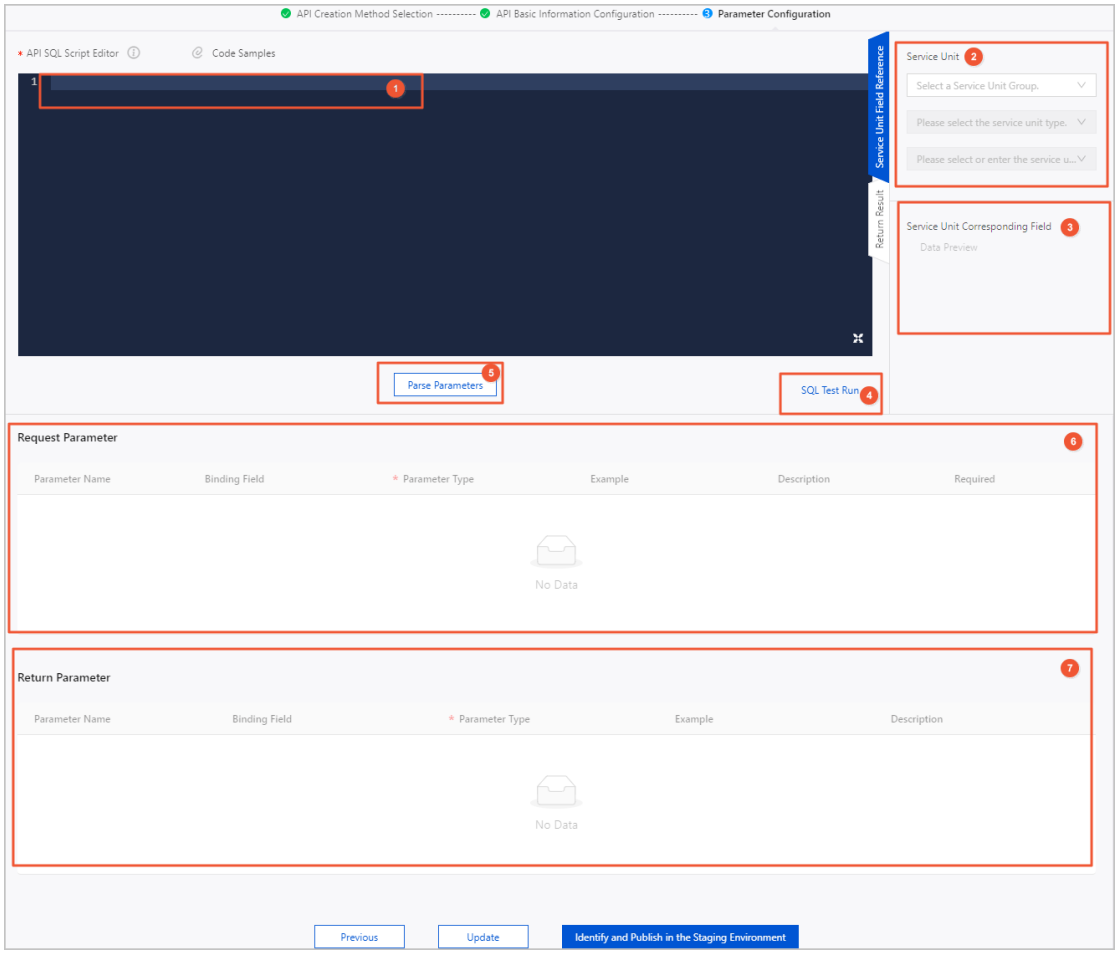
i. In the **API Basic Information Configuration** step, set the parameters as required

Parameter	Description
API Name	The name of the API operation. The name can contain letters, digits, and underscores (_).
Request Method	The method for calling the API operation. Valid values: ■ **GET** ■ **LIST**
Data Update Frequency	The frequency at which data is synchronized by using the API operation. Valid values: ■ **Daily** ■ **Hourly** ■ **Minutely**
Description	The description of the API operation.
Protocol	The protocol that is used by the API operation to synchronize data. Valid values: ■ **HTTP** ■ **HTTPS**

ii. Click **Next Step**.

6. Configure the request and response parameters for the API operation.

i. In the **Parameter Configuration** step, enter an SQL statement in the script editor and set the parameters as required.



No.	GUI element	Description
1	API SQL Script Editor	Enter an SQL statement in the **API SQL Script Editor** section. For example, you can enter the following statement: `select user_id,pid,city_id,activity_time from ads1578301726 where user_id=${user_id};` You can click the icon to view the limits on the SQL statement.
2	Service Unit	To specify a service unit, perform the following steps: a. Select a service unit group from the drop-down list. b. Select a service unit type from the drop-down list. c. Select a service unit from the drop-down list.

No.	GUI element	Description
3	Data Preview	Click **Data Preview** to preview the data of the selected service unit.
4	SQL Test Run	To run the SQL statement, perform the following steps: a. Click **SQL Test Run**. b. In the **Specify Request Parameters** dialog box, set **Test Run Input** to the value that you obtained when you previewed the data of the selected service unit. In this example, enter the value of the `user_id` field. c. Click **Confirm**. ■ If **SQL Status** appears as **Success**, the SQL syntax, computing engine, and database engine passed the validation checks. ■ If **SQL Status** appears as **Error**, the SQL syntax, computing engine, and database engine failed the validation checks.
5	Parse Parameters	Click **Parse Parameters** for the system to automatically generate request and response parameters based on the SQL statement that you entered in the **API SQL Script Editor** section.
6	Request parameters	Specify the following fields for the request parameters: ■ **Parameter Type** ■ **Example** ■ **Description** ■ **Required**
7	Return Parameter	Specify the following fields for the response parameters: ■ **Parameter Type** ■ **Example** ■ **Description**

ii. Click **Identify and Publish in the Staging Environment** to create and publish the API operation in the staging environment.

9.16.5.4. Test an API operation

You can publish an API operation to the production environment only after it passes the test in the staging environment. This topic describes how to test an API operation.

Prerequisites

An API operation is created. For more information, see Create an API operation in template wizard mode and Create an API operation in custom SQL mode.

Context

You can test API operations only in the staging environment. We recommend that you test API operations before you publish them to the production environment.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.
3. On the **APIs** page, find the API operation that you want to test.
   - You can filter API operations by selecting a service unit or service unit group from the corresponding drop-down list.
   - Alternatively, you can enter a keyword in the search box to filter API operations whose names contain the keyword.
4. Move the pointer over the icon in the Actions column and select **Test**.
5. On the **API Test** page, enter the input values for API-specific request parameters based on the example values in the **API Request Parameters** section. Enter the input values for common request parameters based on the example values in the **Public Request Parameter List** section. Select response parameters as required in the **Optional Response Parameters** section. Then, select a protocol from the **Protocol** drop-down list.
6. Click **Test**.
7. After the test is completed, view the test results of the API operation in the **Test Results** section. You can also click **View Error Code Table** in the lower part of the page to view error codes and their descriptions.

9.16.5.5. Publish an API operation

API operations can be called only after they are published to the production environment. The API Service page lists the API operations that have been published. This topic describes how to publish an API operation.

Prerequisites

An API operation is created. For more information, see Create an API operation in template wizard mode and Create an API operation in custom SQL mode.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.

ii. On the Service page, click **Workbench** in the top navigation bar.

iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.

3. On the **APIs** page, find the API operation that you want to publish.
   - You can filter API operations by selecting a service unit or service unit group from the corresponding drop-down list.
   - Alternatively, you can enter a keyword in the search box to filter API operations whose names contain the keyword.
4. Move the pointer over the icon in the Actions column and select **Publish**.
5. In the message that appears, click **OK** to publish the API operation to the production environment.

9.16.5.6. Unpublish an API operation

This topic describes how to unpublish an API operation.

Prerequisites

An API operation is published. For more information, see Publish an API operation.

Context

If you find an API operation inappropriate, you can unpublish it. Unpublished API operations do not appear on the API Service page. After you unpublish an API operation, you can delete it from Development Center. For more information about how to delete an API operation, see Delete an API operation.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.
3. On the **APIs** page, find the API operation that you want to unpublish.
   - You can filter API operations by selecting a service unit or service unit group from the corresponding drop-down list.
   - Alternatively, you can enter a keyword in the search box to filter API operations whose names contain the keyword.
4. Move the pointer over the icon in the Actions column and select **Unpublish**.

> **Note** You can only unpublish API operations that are not being used by any application.

5. In the message that appears, click **OK**.

9.16.5.7. Delete an API operation

This topic describes how to delete an API operation from Development Center.

Prerequisites

An API operation is published. For more information, see Publish an API operation.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Development Center** in the left-side navigation pane.
3. On the **APIs** page, find the API operation that you want to delete.
   - You can filter API operations by selecting a service unit or service unit group from the corresponding drop-down list.
   - Alternatively, you can enter a keyword in the search box to filter API operations whose names contain the keyword.
4. Move the pointer over the icon in the Actions column and select **Delete**.

   > **Note** You can only delete API operations that are not being used by any application.

5. In the message that appears, click **OK**.

9.16.6. Use and manage an API operation

9.16.6.1. Create an application

You must use applications to call API operations in the Services module. An application supports multiple API operations. This topic describes how to create an application.

Prerequisites

An application group is created. For more information, see Create a group.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Applications** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.

iii. On the Workbench page, click **Applications** under **Application Center** in the left-side navigation pane.

3. On the **Applications** page, click **Create Application** in the upper-right corner.
4. In the **Create Application** dialog box, set the **Application Group** and **Application Name** parameters as required.
5. Click **Confirm**.The following table describes the operations that you can perform on the created applications.

Operation	Description
Modify the configuration of an application	Click the icon in the **Actions** column to modify the configuration of an application.
Manage the members of an application	Click the icon in the **Actions** column. The Application Member Management page of the application appears. ○ Click **Add Member**. In the Add Member dialog box, select the name of the account to be added as a member and the user type from the **Member Name** and **Select Role** drop-down lists, respectively. Then, click **Submit** to add a member to the application. ■ The members of an application can apply for permissions on API operations for the application and call the API operations that the application is authorized to access. ■ The administrator of an application can apply for permissions on API operations for the application, call the API operations that the application is authorized to access, and remove API operations from the application. ○ Click the icon in the **Actions** column and select a member from the Transfer Ownership To drop-down list to transfer the ownership of the application to the selected member. ○ Click the icon in the **Actions** column to delete a member from the application. ○ To delete multiple members at a time, select the members and click **Remove** in the upper-right corner.
Go to the API Service page	Click the icon in the **Actions** column. The API Service page appears, where you can view the published API operations.
Delete an application	Click the icon in the **Actions** column to delete an application.
View the key and secret of an application	On the **Applications** page, view the key and secret of an application in the **AppKey** and **AppSecret** columns, respectively.

9.16.6.2. View an API operation

Before you apply for permissions on an API operation for an application, we recommend that you view the details of the API operation and determine whether it meets your business requirements. This topic describes how to view an API operation on the API Service page.

Prerequisites

An API operation is published. For more information, see Publish an API operation.

Procedure

1. Log on to the Dataphin console.
2. Go to the **API Service** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **API Service** in the top navigation bar.
3. On the **API Service** page, find the API operation that you want to view and click **API Documentation** in the **Documentation** column.
4. On the **API Documentation** page, view the information in the **Basic Information**, **API Request Parameters**, **Public Request Parameter List**, and **Response Parameters** sections and determine whether the API operation meets your business requirements.

9.16.6.3. Apply for permissions on an API operation

Before you call an API operation by using an application, the application must be authorized to access the API operation. This topic describes how to apply for the permission to query fields of an API operation for an application.

Prerequisites

An application is created. For more information, see Create an application.

Procedure

1. Log on to the Dataphin console.
2. Go to the **API Service** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **API Service** in the top navigation bar.
3. On the API Service page, find the API operation on which you want to apply for permissions and click **Request Now** in the **Request Status** column.
4. In the Select Application and API step of the **API Permission Request** wizard, set the **Application** and **API** parameters as required.
5. Click **Next**.
6. In the **Apply for API Permissions** step, set the parameters as required.

Section	Parameter	Description

Section	Parameter	Description
Object Information	N/A	In this section, you can view the names of the application and API operation, and the groups to which they belong.
Authorized Fields	N/A	Select the fields on which you want to apply for permissions. You can enter a keyword in the search box to search for the fields whose names contain the keyword.
Permission Configuration	Validity Period	The validity period of the query permission. Valid values: ○ **30 Days** ○ **90 Days** ○ **180 Days** ○ **Long-term** ○ **Custom Expiration Date** Dataphin allows only application users of an application to apply for the permission to query fields of an API operation for the application.
	Reason for Application	The reason for applying for the permission to query fields of the API operation.

7. Click **OK**.

9.16.6.4. Debug an API operation

Before you call an API operation by using an application, you must debug the API operation to ensure data security and stability. This topic describes how to debug an API operation.

Prerequisites

- An application is granted the query permission on the API operation that you want to debug. For more information, see Apply for permissions on an API operation.
- The key and secret of the application that is granted the query permission are obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **APIs** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Workbench** in the top navigation bar.
   iii. On the Workbench page, click **APIs** under **Application Center** in the left-side navigation pane.
3. Find the API operation that you want to debug and click the icon in the **Actions** column.

   The **API Debugging** page appears.
4. On the **API Debugging** page, set the parameters as required.

Section	Parameter	Description
API Request Parameters	Input Value	The value of the required request parameter for debugging the API operation.
Public Request Parameter List	Input Value	The value of the common request parameter for debugging the API operation. ○ **Input Value of appkey**: the key of the application for debugging the API operation. ○ **Input Value of appsecret**: the secret of the application for debugging the API operation.
Response Parameters	Protocol	In the Response Parameters section, select the response parameters and the protocol to be used for the debugging as required.

5. Click **Debug**.
6. After the debugging is completed, view the result in the **Results** section. You can also click **View Error Code Table** in the lower part of the page to view error codes and their descriptions.If the debugging fails, you can go to the APIs page to remove the API operation from the application. To remove an API operation, find the API operation and click the icon in the **Actions** column. In the message that appears, click **OK**. You can also apply for permissions on specific fields of an API operation on the APIs page. To apply for field permissions, find the API operation and click the icon in the **Actions** column. On the **API Permission Request** page, set the parameters as required and click OK.

9.16.6.5. Call an API operation

This topic describes how to use an application to call an API operation that has been published to the production environment.

Prerequisites

- The network is configured. For more information, see Configure the network.
- An application is granted the query permission on an API operation. For more information, see Apply for permissions on an API operation.
- The key and secret of the application are obtained. For more information, see Create an application.
- The API operation is debugged. For more information, see Debug an API operation.

Context

You can call an API operation by using Services SDK for Java or Postman. We recommend that you use Services SDK for Java to call an API operation. You can use Postman only when you debug or develop an API operation.

Procedure

1. Log on to the Dataphin console.

2. Go to the **Call Examples** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the **Service** page, click **Administration** in the top navigation bar.
   iii. On the **Administration** page, click **Call Examples** in the left-side navigation pane.
3. On the **API Call Examples** tab, view the sample code for using Services SDK for Java to call an API operation.
4. Click **Download SDK** in the upper-right corner to download the code package.
5. Modify the downloaded code based on the instructions on the **API Call Examples** tab. Then, you can call the API operation by using the specified application.

9.16.7. Monitor an API operation

Dataphin monitors API requests on a daily basis. You can view the statistics of the API operations that you have published on the Monitoring page. This topic describes how to view the request details of, set a request limit on, and configure alert rules for an API operation.

View the request details of an API operation

1. Log on to the Dataphin console.
2. Go to the **Monitoring** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Administration** in the top navigation bar.
   iii. On the Administration page, click **Monitoring** in the left-side navigation pane.
3. On the **Monitoring** page, click **Details** in the **Actions** column of the API operation whose details you want to view. On the API Monitoring page, you can view the statistics of the API requests in the last hour.

Set a request limit on an API operation

> **Note** To set a request limit on an API operation, you must be the developer of the API operation or the super administrator.

1. On the **Monitoring** page, click **Limit** in the **Actions** column of the API operation on which you want to set a request limit.
2. In the **Configure API Call Limit** dialog box, set the parameters as required.

GUI element	Description
Limit Status	Specifies whether to set a request limit on the API operation.
Time Unit	The time granularity of the request limit to be set on the API operation. Valid values: ○ **hour** ○ **sec** ○ **min** ○ **day**

GUI element	Description
API Call Limit	The maximum number of API requests that can be handled at the specified granularity. Assume that your enterprise has purchased the default specification for the Services module, which supports handling 500 API requests per second. In this case, if you set **Time Unit** to **sec**, you must specify a value less than or equal to 500 for the **API Call Limit** parameter.
Application Call Limit	The maximum number of API requests sent by each involved application that can be handled at the specified granularity. Note the following rules when you set the **Application Call Limit** parameter: ○ The specified value of **Application Call Limit** applies to all the applications that are authorized to access the API operation. ○ The value of **Application Call Limit** must be less than or equal to that of **API Call Limit**. ○ The **API Call Limit** parameter has a higher priority than the **Application Call Limit** parameter. Assume that the request limit on an API operation is 1,000 requests per second and that on each of the three applications that are authorized to access the API operation is 500 requests per second. The sum of the permitted requests per second for the three applications exceeds the request limit on the API operation. However, with a higher priority, the request limit on the API operation determines the upper limit of requests. ○ If you set **Application Call Limit** to a value greater than that of **API Call Limit**, an error message is returned.
Add Application	Click **Add Application** and set the **Application** and **Call Limit** parameters to set a different request limit on a specific application. You can also click the  icon in the **Actions** column of an application to remove the request limit from the application.Note the following rules when you set the **Call Limit** parameter: ○ The value of **Call Limit** for a specific application must be less than or equal to that of **API Call Limit**. ○ The **Call Limit** parameter of a specific application has a higher priority than the **Application Call Limit** parameter. Assume that the request limit on a specific application is 0 or 200 API requests per second and that on each of the other applications that are authorized to access the same API operation is 100 API requests per second. In this case, with a higher priority, the request limit on the specific application determines the upper limit of requests for itself. ○ If you set the **Call Limit** parameter of a specific application to a value greater than that of **API Call Limit**, an error message is returned.

3. Click **OK**.

Configure alert rules for an API operation

1. On the **Monitoring** page, click **Alert** in the **Actions** column of the API operation for which you want to configure alert rules.

2. On the **Alert** page, click **Create Rule** in the upper-right corner.
3. Set the parameters as required.

Parameter	Description
Monitoring Metric	The metric to be monitored for the API operation. Valid values: ○ **Number of Calls**: the total number of API requests per unit time. ○ **Average Response time**: the average response time of API requests per unit time. The average response time equals the total time for processing the API requests divided by the total number of API requests. ○ **Failed Calls (Percentage)**: the ratio of the number of failed API requests to the total number of API requests per unit time. ○ **Calls During Offline (Percentage)**: the ratio of the number of API requests that failed due to connection interruptions to the total number of API requests per unit time.
Time Unit	The time interval at which monitoring data is queried. Valid values: **1min**, **5min**, **10min**, **30min**, and **60min**.
Operator	The comparison operator that is used to compare the value of the monitored metric with the specified value of **Trigger Threshold**. Valid values: <=, <, >, >=, =, != .
Trigger Threshold	The threshold cross which an alert will be triggered.
Notification	The method that is used to send alert notifications to the recipient. Valid values: ○ **Email** ○ **Cellphone** ○ **SMS** ○ **DingTalk**
Recipient	The user to receive alert notifications.
Silent Period	The period of time in which an alert notification is not sent repeatedly. This helps avoid notification redundancy.

4. Click **OK**.On the Alert page, you can view the basic information about the selected API operation and the alert rules that have been created for the API operation. You can perform the following operations on existing alert rules:
   - Click the  icon in the **Actions** column of an alert rule to modify the rule.
   - Click the  icon in the **Actions** column of an alert rule to delete the rule.

9.16.8. Create a Dataphin data source

In the Services module, you can create Dataphin data sources from logical or physical tables for applications to access. This topic describes how to create a Dataphin data source.

Context

After you create Dataphin data sources from logical or physical tables, you can further develop these data resources based on the specific business scenarios. This allows you to enjoy a secure, stable, and easy-to-use data sharing service at low costs.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Dataphin Data Sources** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Administration** in the top navigation bar.
   iii. On the Administration page, click **Dataphin Data Sources** in the left-side navigation pane.
3. On the **Dataphin Data Sources** page, click **Create Data Source** in the upper-right corner.
4. On the **Create Data Source** page, set the parameters as required.

Section	Parameter	Description
Basic Information	Data Source Name	The name of the data source to create.
	Data Source Description	The description of the data source.
Logical Table Selection	Unselected Logical Tables	Lists the logical tables that you are authorized to access. To specify the tables for creating the data source, select the tables and click the > icon to move the tables to the Selected Logical Tables section.
	Selected Logical Tables	Lists the logical tables that you specify for creating the data source. To remove tables from the list, select the tables and click the < icon to move the tables to the Unselected Logical Tables section.
Physical Table Selection	Unselected Physical Tables	Lists the physical tables that you are authorized to access. To specify the tables for creating the data source, select the tables and click the > icon to move the tables to the Selected Logical Tables section.
	Selected Physical Tables	Lists the physical tables that you specify for creating the data source. To remove tables from the list, select the tables and click the < icon to move the tables to the Unselected Physical Tables section.

Section	Parameter	Description
Computing Space	Select Project for Data Source Computing	The Dataphin project for the computing work of the data source to be created.

5. Click **Create Data Source**. After the data source is created, it appears on the Dataphin Data Sources page. This page lists the data sources that applications can use.The following table describes the operations that you can perform on the created data sources.

Operation	Description
Modify the configuration of a data source	Click the icon in the **Actions** column to modify the configuration of a data source.
Delete a data source	Click the icon in the **Actions** column to delete a data source.

9.16.9. Use and manage a Dataphin data source

9.16.9.1. View a Dataphin data source

Before you use a Dataphin data source, we recommend that you view the details of the data source and determine whether it meets your business requirements. This topic describes how to view a Dataphin data source.

Prerequisites

A Dataphin data source is created. For more information, see Create a Dataphin data source.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Dataphin Data Sources** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Administration** in the top navigation bar.
   iii. On the Administration page, click **Dataphin Data Sources** in the left-side navigation pane.
3. On the Dataphin Data Sources page, find the data source that you want to view and click the icon in the **Actions** column. The **View Data Source** page appears.
4. On the **View Data Source** page, view the information in the **Basic Information**, **Logical Table Information**, **Physical Table Information**, and **Computing Space Information** sections and determine whether the data source meets your business requirements.

9.16.9.2. Use a Dataphin data source

Dataphin provides the JDBC interface for you to use Dataphin data sources. You must download the JAR package of the Dataphin JDBC driver and import the JAR package by using a JDBC client or by writing Java code. Then, you can use Dataphin data sources. This topic describes how to use a Dataphin data source.

Prerequisites

- A Dataphin data source that meets your business requirements is found. For more information, see View a Dataphin data source.
- The network is configured. For more information, see Configure the network.
- The permission to query the Dataphin data source is granted by the super administrator.
- The JDBC URL of the Dataphin data source is granted by the super administrator.
- The **AccessKey** of the Dataphin data source is obtained.

Procedure

1. Log on to the Dataphin console.
2. Go to the **Call Examples** page.
   i. On the Dataphin homepage, click **Services** in the top navigation bar.
   ii. On the Service page, click **Administration** in the top navigation bar.
   iii. On the Administration page, click **Call Examples** in the left-side navigation pane.
3. On the **Call Examples** page, click the **Dataphin Data Source Usage Examples** tab.
4. On the Dataphin Data Source Usage Examples tab, click **Download JDBC JAR Package** in the upper-right corner to download the Dataphin JDBC driver.
5. Use a JDBC client or write Java code based on the instructions on the **Dataphin Data Source Usage Examples** tab to use the data source.

10.Elasticsearch (on ECS)

10.1. What is Elasticsearch?

Elasticsearch is a distributed search and data analytics service based on Lucene. It provides a distributed multi-tenant search engine that supports full text queries. This engine is based on a RESTful Web interface. Elasticsearch is developed based on Java. It is released as an open source product that complies with the Apache license terms and conditions. Elasticsearch is a mainstream search engine for enterprises. Elasticsearch is designed to serve cloud computing for real-time search. It is stable, reliable, fast, and easy to install and use.

Apsara Stack Elasticsearch provides two open source versions: Elasticsearch V5.5.3 and Elasticsearch V6.3.2. Apsara Stack Elasticsearch is designed to serve users in data search, data analytics, and other scenarios. Based on open source Elasticsearch, Apsara Stack Elasticsearch also supports enterprise-class permission management.

The default plug-ins provided by Apsara Stack Elasticsearch include but are not limited to the following:

- **IK analyzer**: an open source and lightweight Chinese analysis kit based on Java. The IK analyzer plug-in is very popular in open source communities for Chinese tokenization.
- **Smart Chinese analysis plug-in**: the default Lucene Chinese tokenizer.
- **ICU analysis plug-in**: a Lucene ICU tokenizer. ICU is a set of stable, tested, powerful, and easy to use libraries, providing Unicode and globalization support for applications.
- **Japanese (Kuromoji) analysis plug-in**: a Japanese tokenizer.
- **Stempel (Polish) analysis plug-in**: a French tokenizer.
- **Mapper attachments type plug-in**: an attachment-type plug-in which can parse files of different types into strings based on the Tika library.

10.2. Quick start

This topic shows you how to create an Elasticsearch instance based on ECS. It covers creating a VPC, creating a security group, creating an ECS instance, creating an Elasticsearch instance, and connecting to an Elasticsearch instance.

Create an Elasticsearch instance based on ECS

After you create an Elasticsearch instance, a VPC, and an ECS instance (in the same region as the Elasticsearch instance), the ECS instance can be used as the client. Then you can deploy a user program or run the curl command.

> **Note** If your Elasticsearch instance and ECS instance share the same VPC and region but reside in different zones, you must create a VSwitch in the zone where the ECS instance resides to ensure that the ECS instance can connect to your Elasticsearch instance.

10.2.1. Log on to the Elasticsearch console

This topic describes how to log on to the Elasticsearch console.

Prerequisites

.

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products** > **Big Data** > **Elasticsearch**.
5. Specify **Organization** and **Region** and click **Elasticsearch**.

10.2.2. Create an Elasticsearch cluster

This topic describes how to create an Elasticsearch cluster.

Prerequisites

- A Virtual Private Cloud (VPC) and a VSwitch are created.

  For more information, see the "Quick start" topic in Apsara Stack Virtual Private Cloud User Guide.
- Elastic Compute Service (ECS) instances are sufficient.

Precautions

Before you create an Elasticsearch cluster, you must understand the main data sources of the Elasticsearch cluster. This helps you better plan the size and allocation of disk space.

If you specify a small disk size, Elasticsearch cluster logs occupy a relatively large proportion of disk space. An Elasticsearch cluster mainly stores the following types of data:

- User data that has been pushed to the Elasticsearch cluster.
- Index replicas. You can specify the number of replicas for each index but must make sure that each index has a minimum of one replica.

Procedure

1. Log on to the Elasticsearch console.
2. On the page that appears, click **Create**.
3. Specify the required parameters.

Type	Parameter	Description
Region	Region	The region where the cluster is deployed.
	Zone	An Elasticsearch cluster can be deployed across zones. You may find multiple available zones from the **Zone** drop-down list. Select one or more appropriate zones as needed.
Cluster	Version	Valid values: 5.5.3 and 6.3.2.
	Network Type	The value of this parameter can only be VPC.
	VPC	Select the same VPC as your ECS instances.
	VSwitch	Select the same VSwitch as your ECS instances.
	Specification Family	Valid values: Local SATA, Local SSD, and Cloud Disk.
	Instance Specification	◦ Local SATA 16-Core 64 GB and 8-Core 32 GB ◦ Local SSD 16-Core 64 GB and 8-Core 32 GB ◦ Cloud Disk 16-Core 64 GB, 8-Core 32 GB, and 4-Core 16 GB
	Count	The number of data nodes. You must purchase a minimum of two data nodes. A cluster that contains only two data nodes has a high risk of split-brain. Therefore, exercise caution when you set this parameter.
	Dedicated Master Node	To improve the stability of your cluster, we recommend that you purchase dedicated master nodes. Dedicated master nodes support the following specifications: 4-Core 16 GB, 8-Core 32 GB, and 16-Core 64 GB. Standard SSDs are supported for storage.
	Client Node	For CPU-intensive services, we recommend that you purchase client nodes to share the CPU overheads of data nodes. This further improves the computing performance and service stability of your cluster. For example, if a large number of aggregation operations are performed, you can use client nodes to share the overheads. Client nodes support the following specifications: 4-Core 16 GB, 8-Core 32 GB, and 16-Core 64 GB. Ultra disks are supported for storage.

Type	Parameter	Description
	Warm Node	If your business includes both of the following index types, we recommend that you purchase warm nodes to implement the hot-warm architecture. This architecture improves the computing performance and service stability of your cluster. ◦ Frequently queried or written indexes ◦ Infrequently queried or written indexes, typically indexes of records Warm nodes support the following specifications: 4-Core 16 GB, 8-Core 32 GB, and 16-Core 64 GB. Ultra disks are supported for storage. You can select Warm Node on the buy page or configuration upgrade page to purchase warm nodes. After you purchase nodes, the system adds `-Enode.attr.box_type` to their startup parameters. ◦ Data nodes: -Enode.attr.box_type=hot ◦ Warm nodes: -Enode.attr.box_type=warm
Storage Note This parameter can be configured only when the Specification Family parameter is set to Cloud Disk.	Disk Type	Standard SSDs and ultra disks are supported. ◦ A standard SSD provides a maximum of 2,048 GiB of storage space. Standard SSDs are ideal for online data analytics and searches that require high IOPS and fast responses. ◦ An ultra disk provides a maximum of 5,120 GiB of storage space. Ultra disks are cost-effective and are ideal for scenarios such as logging and analyzing large amounts of data. Ultra disks with the storage space greater than 2,560 GiB cannot be resized because these disks are designed to run in disk arrays or RAID 0.
	Storage Space per Data Node	The storage space of each data node. It depends on the disk type. Unit: GiB. If the disk type is SSD Cloud Disk, the maximum value of this parameter is 2048. If the disk type is Ultra Disk, the maximum value of this parameter is 5120. If the volume of the data that you want to store exceeds 2,048 GiB, you can set this parameter to 2560, 3072, 3584, 4096, 4608, or 5120. If the storage space of the disk for a purchased cluster is less than 2,048 GiB, you can resize the disk to a maximum of 2,048 GiB.
	Username	The value of this parameter can only be elastic.

Password Type	Parameter	Description
	Password	The password of the Elasticsearch cluster. This password is also used to log on to the Kibana console.

4. Click **Create**.

   You can then find the cluster in the cluster list. When **Active** is displayed for the cluster, the cluster is created.

10.2.3. Access an Elasticsearch cluster

This topic describes how to access an Elasticsearch cluster. You can access an Elasticsearch cluster from an Elastic Compute Service (ECS) instance or the Kibana console.

Prerequisites

You have completed the following operations:

- Create an Elasticsearch cluster.
- Create an ECS instance that resides in the same Virtual Private Cloud (VPC), region, and zone as the Elasticsearch cluster.
- Obtain the internal endpoint and Kibana console URL of the Elasticsearch cluster.

  You can log on to the Elasticsearch console, find the target cluster on the Instances page, and click its ID in the **Instance ID/Name** column. Then, obtain the preceding information from the **Basic Information** page.

Use an ECS instance to access an Elasticsearch cluster

1. Log on to the ECS instance over SSH and install the cURL tool.

   > **Note** For information about other methods that are used to log on to an ECS instance, see the "Quick start" topic in Elastic Compute Service (ECS) of Alibaba Cloud Apsara Stack Enterprise V3.12.0 User Guide - Cloud Essentials and Security.

2. Run the following command to connect to the internal endpoint of the Elasticsearch cluster:

   ```
   curl http://<HOST>:<PORT>
   ```

   - Set `<HOST>` to the internal endpoint of the Elasticsearch cluster. You can obtain the internal endpoint from the Basic Information page of the cluster.
   - Set `<PORT>` to the port number of the Elasticsearch cluster. You can obtain the port number from the Basic Information page the cluster. The default port is 9200.

   If the connection is established, the result shown in the following figure is returned.

```
[root@iZ7t                    ~]# curl http://es-cn-d                    .elasticsea
rch.aliyun-inc.com:9200
{
  "name" : "k7d3r4E",
  "cluster_name" : "es-cn-d                    ",
  "cluster_uuid" : "Fmo                    ",
  "version" : {
    "number" : "5.5.3",
    "build_hash" : "9305a5e",
    "build_date" : "2017-09-07T15:56:59.599Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
```

Use the Kibana console to access an Elasticsearch cluster

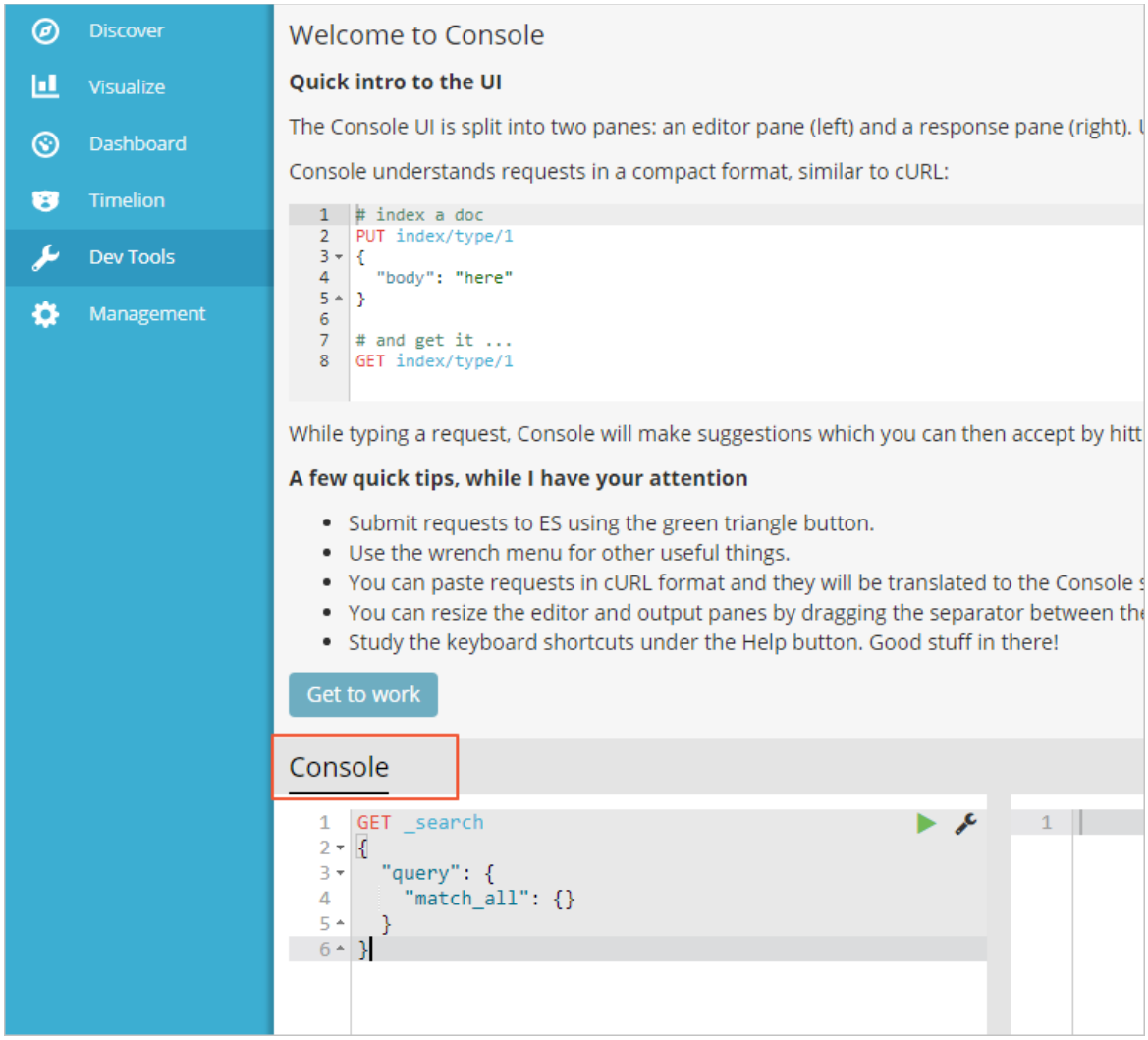
1. Enter the URL of the Kibana console in the address bar of your browser.

   > **Notice** The server where your browser is located must reside in the same VPC as the Elasticsearch cluster.

2. On the page that appears, enter the username and password and click Log in.

   The username is elastic. The password is the one that is specified when you create the Elasticsearch cluster.

3. In the left-side navigation pane, click **Dev Tools**.
4. On the **Console** tab, run the `GET /` command to access the Elasticsearch cluster.



10.3. Manage clusters

This topic describes how to manage Elasticsearch clusters. Elasticsearch provides management features, such as **Kibana console**, **cluster restart**, and **refresh**.

10.3.1. Log on to the Kibana console

This topic describes how to log on to the Kibana console of an Elasticsearch cluster. Elasticsearch provides the Kibana console for you to scale your businesses. The Kibana console is seamlessly integrated into Elasticsearch. It allows you to view the status of your Elasticsearch cluster in real time and manage the cluster.

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. On the **Basic Information** page, obtain the **Kibana console URL**. Then, enter the URL in the address bar of your browser.

> **Notice** The server where your browser is located must reside in the same Virtual Private Cloud (VPC) as the Elasticsearch cluster.

4. On the page that appears, enter the username and password and click Log in.

   The username is elastic. The password is the one that is specified when you create the Elasticsearch cluster.

10.3.2. Restart an Elasticsearch cluster

This topic describes how to restart an Elasticsearch cluster. Two restart methods are supported: Restart and Force Restart. You can select a method based on your business scenario.

Precautions

- If the disk usage of an Elasticsearch cluster exceeds 85%, the status of the cluster may become abnormal (indicated by the color yellow or red). In this case, you must perform a forced restart.
- If the cluster is abnormal, we recommend that you do not perform the following operations on the cluster: node addition, node capacity expansion, disk resizing, restart, password reset, and configuration update. Perform these operations only after the status of the cluster becomes normal (indicated by the color green).
- If you update the configuration of an abnormal cluster that contains two or more nodes and the cluster remains in the Initializing state, submit a ticket.
- If a cluster contains only one node, services may become unavailable when you update the cluster configuration, restart or scale out/in the cluster, or reset the password of the cluster. In this case, create another Elasticsearch cluster and migrate your services to the new cluster.

Procedure

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. On the **Basic Information** page, click **Restart Instance**.
4. In the **Restart Instance** dialog box, select a restart method.
   - **Restart**: This restart method does not affect the services of your cluster but is time-consuming. Before you select this method, make sure that each index of the cluster has a minimum of one replica.
   - **Force Restart**: This restart method may cause unstable services on your cluster but saves time.
5. Click **OK**.

> **Notice** Before the restart, make sure that the cluster is in a normal state. During the restart, the CPU utilization and memory usage of nodes in the cluster experience a surge. This may affect the stability of your services for a short period of time.

10.3.3. Refresh the information of an Elasticsearch cluster

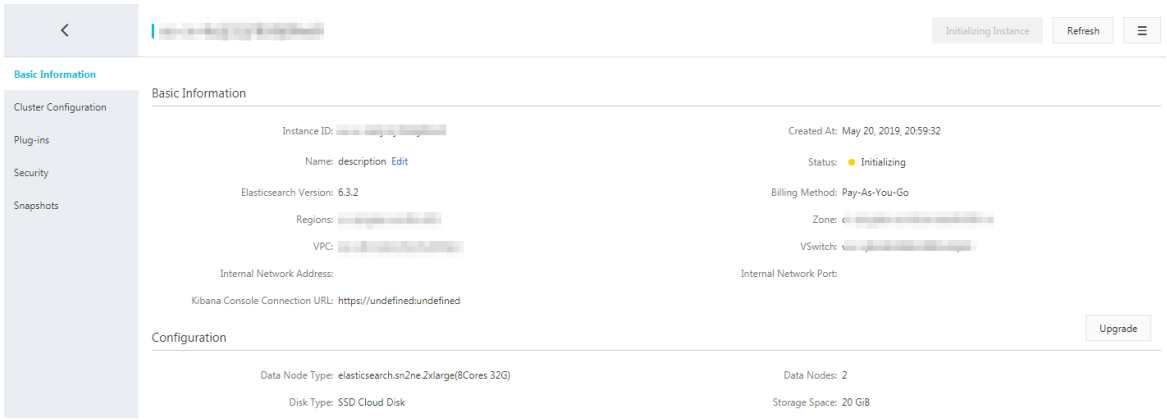
This topic describes how to refresh the information of an Elasticsearch cluster. If part of the information in the console, such as the status of a newly created Elasticsearch cluster, is not refreshed in time, the console may fail to display the information. In this case, you can manually refresh the page to obtain the latest status.

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the upper-right corner of the **Basic Information** page, click **Refresh**.

10.3.4. View the basic information of an Elasticsearch cluster

This topic describes how to view the basic information of an Elasticsearch cluster.

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. On the **Basic Information** page, view the basic information of the cluster.



Parameter	Description
Instance ID	The unique ID of the cluster.
Created At	The time when the cluster was created.
Name	The name of the cluster. By default, the name of a cluster is the same as its ID. Cluster names are configurable. You can search for clusters by name.
Status	The status of the cluster. A cluster has the following states: **Active** (green), **Initializing** (yellow), **Unhealthy** (red), **Paused** (red), and **Expired** (gray).
Elasticsearch Version	The version of the cluster. Valid values: **5.5.3** and **6.3.2**.
Region	The region where the cluster resides.
Zone	The zone where the cluster resides.

Parameter	Description
VPC	The Virtual Private Cloud (VPC) where the cluster resides.
VSwitch	The VSwitch to which the cluster belongs.
Internal Network Address	The internal endpoint of the cluster. You can use an Elastic Compute Service (ECS) instance to connect to the internal endpoint of the cluster. Make sure that the ECS instance resides in the same VPC as the cluster.
Internal Network Port	The port used to connect to the cluster over an internal network. Elasticsearch supports the following ports: ○ Port 9200 for HTTP ○ Port 9300 for TCP
Kibana Console Connection URL	The URL of the Kibana console of the cluster. You can use an ECS instance to connect to the URL. Make sure that the ECS instance resides in the same VPC as the cluster.

Note You can also view the cluster configuration information on the Basic Information page. The information includes the data node specifications, number of data nodes, disk type, and storage space. You can determine whether to upgrade the cluster configuration based on the information and your business plan.

10.3.5. Upgrade the configuration of an Elasticsearch cluster

Apsara Stack Elasticsearch allows you to upgrade the specifications, storage space per node, and number of nodes for an Elasticsearch cluster. You cannot downgrade the configuration of an Elasticsearch cluster. This topic describes how to upgrade the configuration of an Elasticsearch cluster.

Procedure

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the lower-right corner of the **Basic Information** page, click **Upgrade**.
4. On the **Update Configuration** page, modify the cluster configuration.

   For information about parameters on this page, see Create an Elasticsearch cluster.

> Notice
>
> - For each upgrade, you can modify only one of the following items: specifications, storage space per node, and the number of nodes.
> - If your business requires a cluster configuration upgrade, evaluate your Elasticsearch cluster before the upgrade.

5. Click **Confirm Change**.

10.3.6. Configure an Elasticsearch cluster

10.3.6.1. Configure synonyms

You can configure synonyms for the synonym dictionary of an Apsara Stack Elasticsearch cluster. After you configure a tokenizer, new indexes in the cluster are tokenized based on the latest synonym dictionary. This topic describes how to configure synonyms.

Description

> Notice
>
> - After you upload a synonym dictionary file to an Elasticsearch cluster, the system does not restart the cluster. The system sends the synonym dictionary file to all nodes in the cluster. The time that is required for the file to take effect depends on the number of nodes in the cluster.
> - For example, the index-aliyun index is created based on the aliyun.txt synonym dictionary file. If you have uploaded a new synonym dictionary file to overwrite the existing aliyun.txt file, the index-aliyun index cannot automatically load the new dictionary file. If you want the index to automatically load the new dictionary file, we recommend that you disable the index before the upload and then enable the index after the upload. If an index that is created before the uploaded synonym dictionary file takes effect needs to use synonyms, you must reindex the data in the index and configure synonyms.

You can use a filter to configure synonyms. Sample code:

```
PUT /test_index
{
  "settings": {
    "index" : {
      "analysis" : {
        "analyzer" : {
          "synonym" : {
            "tokenizer" : "whitespace",
            "filter" : ["synonym"]
            }
          },
          "filter" : {
            "synonym" : {
              "type" : "synonym",
               "synonyms_path" : "analysis/synonym.txt",
               "tokenizer" : "whitespace"
             }
           }
         }
        }
     }
}
```

- `filter` : specifies a `synonym` filter that contains the `analysis/synonym.txt` path. This path indicates the location of config.
- `tokenizer` : specifies a tokenizer that tokenizes synonyms. It is set to `whitespace` by default. Additional settings:
  - `ignore_case` : The default value is false.
  - `expand` : The default value is true.

Two synonym formats are supported: Solr and WordNet.

- Solr synonyms

  Configuration example:

```
# Blank lines and lines starting with pound are comments.
# Explicit mappings match any token sequence on the LHS of "=>"
# and replace with all alternatives on the RHS.  These types of mappings
# ignore the expand parameter in the schema.
# Examples:
i-pod, i pod => ipod,
sea biscuit, sea biscit => seabiscuit
# Equivalent synonyms may be separated with commas and give
# no explicit mapping.  In this case the mapping behavior will
# be taken from the expand parameter in the schema.  This allows
# the same synonym file to be used in different synonym handling strategies.
# Examples:
ipod, i-pod, i pod
foozball , foosball
universe , cosmos
lol, laughing out loud
# If expand==true, "ipod, i-pod, i pod" is equivalent
# to the explicit mapping:
ipod, i-pod, i pod => ipod, i-pod, i pod
# If expand==false, "ipod, i-pod, i pod" is equivalent
# to the explicit mapping:
ipod, i-pod, i pod => ipod
# Multiple synonym mapping entries are merged.
foo => foo bar
foo => baz
# is equivalent to
foo => foo bar, baz
```

You can also define synonyms in the filter, but you must use `synonyms` rather than `synonyms_path`. Example:

```
PUT /test_index
{
  "settings": {
    "index" : {
      "analysis" : {
        "filter" : {
          "synonym" : {
            "type" : "synonym",
            "synonyms" : [
              "i-pod, i pod => ipod",
              "begin, start"
            ]
          }
        }
      }
    }
}
```

We recommend that you use `synonyms_path` to define large synonym sets in the filter. Using `synonyms` to define large synonym sets increases the size of your cluster.

- WordNet synonyms

  Configuration example:

```
PUT /test_index
{
  "settings": {
    "index" : {
      "analysis" : {
        "filter" : {
          "synonym" : {
            "type" : "synonym",
            "format" : "wordnet",
            "synonyms" : [
              "s(100000001,1,'abstain',v,1,0).",
              "s(100000001,2,'refrain',v,1,0).",
              "s(100000001,3,'desist',v,1,0)."
            ]
          }
        }
      }
    }
  }
}
```

You can also use `synonyms_path` to define WordNet synonyms.

Use a synonym dictionary file to configure synonyms

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane of the page that appears, click **Cluster Configuration**.
4. Upload a synonym dictionary file.
   i. Click **Synonym Dictionary Configuration**.
   ii. In the **Synonym Dictionary Configuration** pane, select the mode to upload a synonym dictionary file and click **Upload**. Then, select the .txt file that is generated based on the preceding rules.
   iii. Click **Save**.

   After the state of the Elasticsearch cluster becomes Active, you can use the synonym dictionary file. In this example, the `aliyun_synonyms.txt` file is uploaded. The file contains `begin, start` .
5. Configure and test the synonym dictionary.
   i. Log on to the Kibana console.

   For more information, see Log on to the Kibana console.
   ii. In the left-side navigation pane, click **Dev Tools**.

iii. On the **Console** tab, run the following command to create an index:

```
PUT aliyun-index-test
{
"index": {
 "analysis": {
  "analyzer": {
   "by_smart": {
    "type": "custom",
    "tokenizer": "ik_smart",
    "filter": ["by_tfr","by_sfr"],
    "char_filter": ["by_cfr"]
   },
   "by_max_word": {
    "type": "custom",
    "tokenizer": "ik_max_word",
    "filter": ["by_tfr","by_sfr"],
    "char_filter": ["by_cfr"]
   }
  },
  "filter": {
   "by_tfr": {
    "type": "stop",
    "stopwords": [" "]
   },
   "by_sfr": {
    "type": "synonym",
    "synonyms_path": "analysis/aliyun_synonyms.txt"
   }
  },
  "char_filter": {
   "by_cfr": {
    "type": "mapping",
    "mappings": ["| => |"]
   }
  }
 }
}
}
```

iv. Run the following command to configure the title synonym field:

```
PUT aliyun-index-test/_mapping/doc
{
"properties": {
 "title": {
   "type": "text",
   "index": "analyzed",
   "analyzer": "by_max_word",
   "search_analyzer": "by_smart"
 }
}
}
```

v. Run the following command to verify synonyms:

```
GET aliyun-index-test/_analyze
{
"analyzer": "by_smart",
"text":"begin"
}
```

If the command is successfully executed, the following result is returned:

```
{
"tokens": [
 {
   "token": "begin",
   "start_offset": 0,
   "end_offset": 5,
   "type": "ENGLISH",
   "position": 0
 },
 {
   "token": "start",
   "start_offset": 0,
   "end_offset": 5,
   "type": "SYNONYM",
   "position": 0
 }
]
}
```

vi. Run the following commands to add data for further testing:

```
PUT aliyun-index-test/doc/1
{
"title": "Shall I begin?"
}
```

```
PUT aliyun-index-test/doc/2
{
"title": "I start work at nine."
}
```

vii. Run the following command to perform a query test:

```
GET aliyun-index-test/_search
{
 "query" : { "match" : { "title" : "begin" }},
 "highlight" : {
    "pre_tags" : ["<red>", "<bule>"],
    "post_tags" : ["</red>", "</bule>"],
    "fields" : {
       "title" : {}
    }
 }
}
```

If the command is successfully executed, the following result is returned:

```
{
"took": 11,
"timed_out": false,
"_shards": {
 "total": 5,
 "successful": 5,
 "failed": 0
},
"hits": {
 "total": 2,
 "max_score": 0.41048482,
 "hits": [
  {
    "_index": "aliyun-index-test",
    "_type": "doc",
    "_id": "2"
```

```
    "_id": "2",
    "_score": 0.41048482,
    "_source": {
      "title": "I start work at nine."
    },
    "highlight": {
      "title": [
        "I <red>start</red> work at nine."
      ]
    }
  },
  {
    "_index": "aliyun-index-test",
    "_type": "doc",
    "_id": "1",
    "_score": 0.39556286,
    "_source": {
      "title": "Shall I begin?"
    },
    "highlight": {
      "title": [
        "Shall I <red>begin</red>?"
      ]
    }
  }
 ]
}
}
```


Reference synonyms and use the IK dictionary for word splitting

1. Configure the `my_synonym_filter` synonym filter and a synonym dictionary.
2. Configure the `my_synonyms` analyzer and use the `ik_smart` IK analyzer to split words.

   The `ik_smart` IK analyzer splits words and converts all letters into lowercase.

```
PUT /my_index
{
 "settings": {
    "analysis": {
       "analyzer": {
          "my_synonyms": {
             "filter": [
                "lowercase",
                "my_synonym_filter"
             ],
             "tokenizer": "ik_smart"
          }
       },
       "filter": {
          "my_synonym_filter": {
             "synonyms": [
                "begin,start"
             ],
             "type": "synonym"
          }
       }
    }
 }
}
```

3. Run the following command to configure the title synonym field:

```
PUT /my_index/_mapping/doc
{
"properties": {
 "title": {
   "type": "text",
   "index": "analyzed",
   "analyzer": "my_synonyms"
 }
}
}
```

4. Run the following command to verify synonyms:

```
GET /my_index/_analyze
{
 "analyzer":"my_synonyms",
 "text":"Shall I begin?"
}
```

If the command is successfully executed, the following result is returned:

```
{
"tokens": [
 {
  "token": "shall",
  "start_offset": 0,
  "end_offset": 5,
  "type": "ENGLISH",
  "position": 0
 },
 {
  "token": "i",
  "start_offset": 6,
  "end_offset": 7,
  "type": "ENGLISH",
  "position": 1
 },
 {
  "token": "begin",
  "start_offset": 8,
  "end_offset": 13,
  "type": "ENGLISH",
  "position": 2
 },
 {
  "token": "start",
  "start_offset": 8,
  "end_offset": 13,
  "type": "SYNONYM",
  "position": 2
 }
]
}
```

5. Run the following commands to add data for further testing:

```
PUT /my_index/doc/1
{
"title": "Shall I begin?"
}
```

```
PUT /my_index/doc/2
{
"title": "I start work at nine."
}
```

6. Run the following command to perform a query test:

```
GET /my_index/_search
{
"query" : { "match" : { "title" : "begin" }},
"highlight" : {
 "pre_tags" : ["<red>", "<bule>"],
 "post_tags" : ["</red>", "</bule>"],
 "fields" : {
    "title" : {}
 }
}
}
```

If the command is successfully executed, the following result is returned:

```
{
"took": 11,
"timed_out": false,
"_shards": {
 "total": 5,
 "successful": 5,
 "failed": 0
},
"hits": {
 "total": 2,
 "max_score": 0.41913947,
 "hits": [
  {
   "_index": "my_index",
   "_type": "doc",
   "_id": "2"
```

```
    "_id": "2",
    "_score": 0.41913947,
    "_source": {
      "title": "I start work at nine."
    },
    "highlight": {
      "title": [
        "I <red>start</red> work at nine."
      ]
    }
  },
  {
    "_index": "my_index",
    "_type": "doc",
    "_id": "1",
    "_score": 0.39556286,
    "_source": {
      "title": "Shall I begin?"
    },
    "highlight": {
      "title": [
        "Shall I <red>begin</red>?"
      ]
    }
  }
 ]
}
}
```


10.3.6.2. Perform configurations on YML files

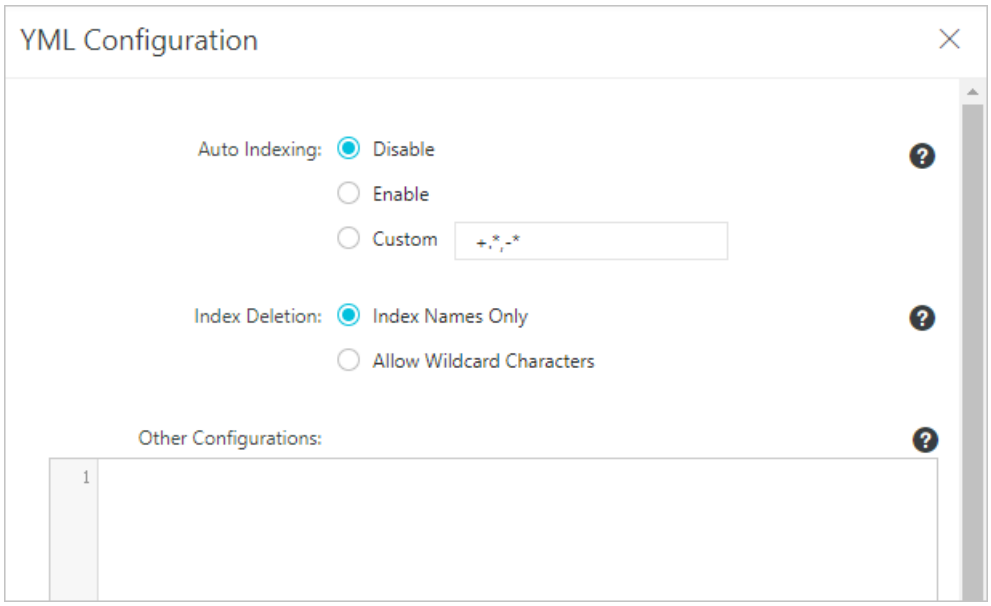
10.3.6.2.1. Configure a YML file

The YML file contains the configuration of an Elasticsearch cluster. You can use the file to modify the configuration of the cluster. This topic describes how to configure a YML file.

Procedure

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane of the page that appears, click **Cluster Configuration**.
4. On the page that appears, click **Modify Configuration** on the right side of **YML Configuration**.

5. In the **YML Configuration** pane, specify the required parameters.



Parameter	Description
Auto Indexing	This parameter specifies whether to automatically create an index when a new document is uploaded to an Elasticsearch cluster but no index has been created. Valid values: ○ Disable: The system does not automatically create an index. This value is the default value. ○ Enable: The system automatically creates an index. **Notice** We recommend that you disable Auto Indexing because indexes created by this feature may not meet your business requirements. ○ Custom: The system automatically creates a custom index. For example, if you specify `-an*,+a,-*`, the system automatically creates only indexes whose names start with a, except indexes whose names start with an.
Index Deletion	○ Index Names Only: When you delete an index, you must specify its name. ○ Allow Wildcard Characters: You can use wildcards to delete multiple indexes at a time. **Notice** You cannot restore the indexes that have been deleted. Exercise caution when you specify this parameter.

Parameter	Description
Other Configurations	The following content lists some supported configuration items. For more information, see YML configuration parameters. ○ http.cors.enabled ○ http.cors.allow-origin ○ http.cors.max-age ○ http.cors.allow-methods ○ http.cors.allow-headers ○ http.cors.allow-credentials ○ reindex.remote.whitelist ○ action.auto_create_index ○ action.destructive_requires_name Notice You can reindex data from a remote Elasticsearch cluster that is specified in a remote reindex whitelist. You can reindex data from all versions of remote Elasticsearch clusters. This feature allows you to migrate index data from an Elasticsearch cluster of an earlier version to an Elasticsearch cluster of the newly released version. For more information, see Reindex data from a remote Elasticsearch cluster.

10.3.6.2.2. YML configuration parameters

This topic describes the YML configuration parameters supported by Elasticsearch. These configuration parameters support only the static configuration mode but not the hot deployment mode. For configurations to take effect, you must add the configurations to the elasticsearch.yml file.

Parameters

Parameter	Description

Parameter	Description
http.cors.enabled	The cross-origin resource sharing (CORS) configuration item. This item is used to specify whether to allow browsers on other origins to access Elasticsearch. • If you set the value to `true`, CORS is enabled, and Elasticsearch can process `OPTIONS CORS` requests. • If the origin in a request is declared in `http.cors.allow-origin`, Elasticsearch returns a response that has the `Access-Control-Allow-Origin` header included. • The default value is `false`. If you set the value to `false`, CORS is disabled. In this case, Elasticsearch ignores the origin in the request header and returns a response that does not have the `Access-Control-Allow-Origin` header included. • If a client cannot send a pre-flight request that has origin information included in the request header or does not verify the `Access-Control-Allow-Origin` header in the response that is returned from the server, the cross-origin security is compromised. • If CORS is disabled for Elasticsearch, a client can send only an OPTIONS request to check whether the Access-Control-Allow-Origin header exists.
http.cors.allow-origin	The origin configuration item. This item specifies the origins from which requests are allowed. By default, no origins are allowed. • If you add a forward slash (`/`) to the start and end of the value, this item is considered as a regular expression. • This allows you to use regular expressions to support HTTP and HTTPS requests. For example, `/https?:\/\/localhost(:[0-9]+)?/` indicates that Elasticsearch responds to all requests that match the regular expression. • The asterisk (`*`) is a valid character but considered as a security risk. This is because the asterisk indicates that an Elasticsearch cluster is open to all origins. We recommend that you do not use asterisks.
http.cors.max-age	Browsers can send OPTIONS requests to query the CORS configuration. This item specifies the cache time of the retrieved CORS configuration. The default value is 1728000 seconds (20 days).
http.cors.allow-methods	The item that is used to configure the request method. Valid values: `OPTIONS, HEAD, GET, POST, PUT, and DELETE`.
http.cors.allow-headers	The item that is used to configure the request header. Valid values: `X-Requested-With, Content-Type, and Content-Length`.

Parameter	Description
http.cors.allow-credentials	The credential configuration item. This item specifies whether Elasticsearch is allowed to return the `Access-Control-Allow-Credentials` header. If you set the value to `true`, Elasticsearch is allowed to return the header. The default value is `false`.
reindex.remote.whitelist	The remote reindex whitelist of the current cluster. This whitelist contains the addresses of remote hosts that can be used to access the current cluster. An address in the whitelist can be a combination of host and port. Separate the configurations of multiple hosts with commas (,), such as `otherhost:9200,another:9200,127.0.10.**:9200,localhost:**`. You can use only the host and port to configure security policies because the whitelist ignores the protocol information.
action.auto_create_index	The configuration item for auto index creation. If you set the value to `false`, the auto index creation feature is disabled.
action.destructive_requires_name	This item specifies whether to specify the name of an index when you delete the index. The default value is `false`. If you set the value to false, you can use a regular expression or the `_all` parameter to delete indexes. If you set the value to `true`, you must specify the names of the indexes you want to delete. In this case, you cannot use `_all` or wildcards.

10.3.6.2.3. Reindex data from a remote Elasticsearch cluster

This topic describes how to reindex data from a remote Elasticsearch cluster that is specified in a remote reindex whitelist. This feature is applicable to all versions of remote Elasticsearch clusters. It allows you to reindex data from an Elasticsearch cluster of an earlier version to an Elasticsearch cluster of the newly released version.

You can call the reindex operation to reindex data. Example:

```
POST _reindex
{
 "source": {
  "remote": {
   "host": "http://otherhost:9200",
   "username": "user",
   "password": "pass"
  },
  "index": "source",
  "query": {
   "match": {
    "test": "data"
   }
  }
 },
 "dest": {
  "index": "dest"
 }
}
```

Note Reindexing from a remote cluster does not support manual slicing or automatic slicing.

- The value of the `host` parameter must include the protocol, domain name, and port number, such as http://otherhost:9200.
- `username` and `password` are optional. If the remote Elasticsearch cluster needs to use the basic authorization scheme, specify the username and password. If you use the basic authorization scheme, we recommend that you use the HTTPS protocol. Otherwise, the password is transmitted as a text.
- The reindex operation can be called remotely only after the remote host address is declared in the *elasticsearch.yaml* configuration file by using the `reindex.remote.whitelist` attribute. `reindex.remote.whitelist` can use host-port pairs. Separate multiple pairs with commas (,), such as, otherhost:9200, another:9200, 127.0.10.**:9200, localhost:**. You can use only the host and port to configure security policies because the whitelist ignores the protocol information.
- If the host is already added to the whitelist, the query is directly sent to the remote Elasticsearch cluster without verification or modification.

Indexing from a remote Elasticsearch cluster uses on-heap buffer. The default maximum batch size is 100 MB. If the index on the remote cluster contains large documents, you must adjust the batch size to a small value.

In the following example, the batch size is 10, which is the minimum value.

```
POST _reindex
{
 "source": {
  "remote": {
   "host": "http://otherhost:9200"
  },
  "index": "source",
  "size": 10,
  "query": {
   "match": {
    "test": "data"
   }
  }
 },
 "dest": {
  "index": "dest"
 }
}
```

- `socket_timeout` is used to set the timeout period for socket read. The default value is 30s.
- `connect_timeout` is used to set the connection timeout period. The default value is 1s.

In the following example, socket_timeout is set to 1m and connect_timeout is set to 10s.

```
POST _reindex
{
 "source": {
  "remote": {
   "host": "http://otherhost:9200",
   "socket_timeout": "1m",
   "connect_timeout": "10s"
  },
  "index": "source",
  "query": {
   "match": {
    "test": "data"
   }
  }
 },
 "dest": {
  "index": "dest"
 }
}
```

10.3.7. Configure plug-ins

Plug-ins extend the capabilities of Apsara Stack Elasticsearch in data pre-processing and analytics. You can use built-in plug-ins or upload custom plug-ins based on your requirements. This topic describes how to configure a plug-in.

Go to the plug-in configuration page

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Plug-ins**.

Built-in plug-ins

The following table lists the built-in plug-ins for Elasticsearch on the Plug-ins page.

Plug-in	Description
analysis-icu	The ICU analysis plug-in for Elasticsearch. This plug-in integrates the ICU module of Lucene into Elasticsearch and adds ICU analysis components.
analysis-ik	The IK analysis plug-in for Elasticsearch.

Plug-in	Description
analysis-kuromoji	The Japanese (Kuromoji) analysis plug-in for Elasticsearch. This plug-in integrates the Kuromoji analysis module of Lucene into Elasticsearch.
analysis-phonetic	The phonetic analysis plug-in for Elasticsearch. This plug-in integrates the phonetic token filter into Elasticsearch.
analysis-pinyin	The pinyin analysis plug-in for Elasticsearch.
analysis-smartcn	The smart Chinese analysis plug-in for Elasticsearch. This plug-in integrates the smart Chinese analysis module of Lucene into Elasticsearch.
elasticsearch-repository-oss	The plug-in allows you to use Apsara Stack Object Storage Service (OSS) to store Elasticsearch snapshots.
gab-paas-plugin	None.
ingest-attachment	The ingest processor for Elasticsearch. This plug-in uses Apache Tika to extract content.
ingest-geoip	The ingest processor for Elasticsearch. This plug-in queries geographic data in MaxMind databases based on IP addresses.
ingest-user-agent	The ingest processor for Elasticsearch. This plug-in extracts information from a user agent.
mapper-murmur3	When you create an index and store these values in the index, this plug-in allows you to compute the hash values of fields.
mapper-size	This plug-in allows you to take note of the sizes of documents before they are compressed when you create an index.
repository-hdfs	This plug-in provides support for Hadoop Distributed File System (HDFS) repositories.
search-guard-6	This plug-in provides the access control feature for Elasticsearch.

The analysis-ik plug-in allows you to use the standard update or rolling update method to update IK dictionaries. For more information, see Standard update of IK dictionaries and Rolling update of IK dictionaries.

Standard update of IK dictionaries

The standard update method requires Elasticsearch to perform a rolling restart for an Elasticsearch cluster to update dictionaries. Elasticsearch sends the uploaded dictionary file to all nodes in the cluster, modifies the `IKAnalyzer.cfg.xml` file, and then restarts the nodes to load the file.

You can use the standard update method to update the IK main dictionary and stopword list. In the standard update pane, you can check the built-in main dictionary `SYSTEM_MAIN.dic` and the built-in stopword list `SYSTEM_STOPWORD.dic`.

- If you want to update the built-in main dictionary, upload a dictionary file named `SYSTEM_MAIN.dic`.
- If you want to update the built-in stopword list, upload a dictionary file named `SYSTEM_STOPWORD.dic`.

1. Go to the plug-in configuration page.
2. On the page that appears, find the analysis-ik plug-in and click **Standard Update** in the Actions column.
3. In the **Plug-ins** pane, click **Configure**.
4. Click **Upload DIC File** and select a local file to upload.

   > Note
   > - By default, a .dic file needs to be uploaded. You can also choose to upload an OSS file.
   > - If the content of a dictionary stored in OSS changes, you must manually upload the dictionary file again.

5. In the lower part of the pane, select **This operation will restart the instance. Continue?** and click **Save**.

   Then, the system performs a rolling start for the Elasticsearch cluster.

6. After the cluster is restarted, log on to the Kibana console of the Elasticsearch cluster and run the following command to check whether the new dictionaries take effect.

   For more information about how to log on to the Kibana console, see Use the Kibana console to access an Elasticsearch cluster.

   ```
   GET _analyze
   {
   "analyzer": "ik_smart",
   "text": ["Tokens in the new dictionaries"]
   }
   ```

   > Note
   > - You cannot delete the built-in main dictionary and stopword list.
   > - The standard update operation requires Elasticsearch to perform a rolling restart for the cluster.
   > - You can perform the standard update operation only when the cluster is healthy.

Rolling update of IK dictionaries

If the content of a dictionary file changes, you can use this method to update the dictionaries on all nodes in an Elasticsearch cluster. After you upload the latest dictionary file, all nodes in the Elasticsearch cluster automatically load the file.

If the dictionary file list changes when you perform a rolling update, the system performs a rolling restart for your Elasticsearch cluster to reload the dictionary configuration. For example, when you upload a new dictionary file or delete an existing dictionary file, the changes are synchronized to the `IKAnalyzer.cfg.xml` file.

The procedure of a rolling update is similar to that of a standard update. If you upload a dictionary file for the first time, you must modify the `IKAnalyzer.cfg.xml` file. Elasticsearch needs to perform a rolling restart to reload the configuration file. Subsequently, if you upload a dictionary file with the same name, Elasticsearch does not perform a rolling restart for the update to take effect.

1. Go to the plug-in configuration page.
2. On the page that appears, find the analysis-ik plug-in and click **Rolling Update** in the Actions column.
3. In the **Plug-ins** pane, click **Configure**.
4. Click **Upload DIC File** and select a local file to upload.

   > **Note**
   > - By default, a .dic file needs to be uploaded. You can also choose to upload an OSS file.
   > - If the content of a dictionary stored in OSS changes, you must manually upload the dictionary file again.

5. In the lower part of the pane, select **This operation will restart the instance. Continue?** and click **Save**.

   After you click Save, the system performs a rolling restart for the cluster. After the cluster is restarted, the new dictionaries take effect.

If you want to add tokens to or remove tokens from the new dictionaries, perform the following steps to modify the `a_10words.dic` file: In the rolling update pane, delete the existing `a_10words.dic` file and upload a new dictionary file. The new dictionary file must have the same name. This operation changes the content of the existing dictionary file in the cluster and uploads a new file that has the same name. Elasticsearch does not need to perform a rolling restart on the cluster for the update to take effect. You can directly click **Save**.

The analysis-ik plug-in on the nodes of the Elasticsearch cluster automatically loads the dictionary file. The time that is required to load the dictionary file varies depending on the nodes. It requires about two minutes for all nodes to load the dictionary file. To verify the new dictionaries, you can log on to the Kibana console of the Elasticsearch cluster and run the following command multiple times:

```
GET _analyze
{
"analyzer": "ik_smart",
"text": ["Tokens in the new dictionaries"]
}
```

> **Note** You cannot use the rolling update method to modify the built-in main dictionary. If you want to modify the built-in main dictionary, you must use the standard update method.

10.3.8. Configure security settings

You can reset the password of your Elasticsearch cluster, modify the Kibana whitelist, and modify the VPC whitelist. This topic describes how to configure security settings.

Go to the security configuration page

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Security**.

Reset the password of an Elasticsearch cluster

> **Notice** This operation only resets the password of the elastic account. We recommend that you do not use the elastic account to log on to your Elasticsearch cluster.

1. Go to the security configuration page.
2. Click **Reset** next to **Elasticsearch Instance Password**.
3. In the **Reset** pane, enter a new password and confirm the password.
4. Click **OK**.

   Then, the new password takes effect within about five minutes. The new password will be used to log on to the Kibana console and access the Elasticsearch cluster.

   > **Note** The password reset does not restart the Elasticsearch cluster.

Modify a Kibana whitelist

1. Go to the security configuration page.
2. Click **Update** next to **Kibana Whitelist**.
3. In the **Edit Kibana Whitelist** pane, enter IP addresses.

   You can enter both IP addresses and CIDR blocks. For example, enter `192.168.0.1` or `192.168.0.0/24`. Separate multiple IP addresses or CIDR blocks with commas (,). You can enter `127.0.0.1` to deny requests from all IPv4 addresses or enter `0.0.0.0/0` to allow requests from all IPv4 addresses.

   If your Elasticsearch cluster is deployed in the China (Hangzhou) region, you can add IPv6 addresses to the whitelist. For example, enter `2401:b180:1000:24::5` or `2401:b180:1000::/48`. Enter `::1` to deny requests from all IPv6 addresses or `::/0` to allow requests from all IPv6 addresses.

> Note You can use only a server that resides in the same VPC as your Elasticsearch cluster to log on to the Kibana console.

4. Click **OK**.

Modify a VPC whitelist

1. Go to the security configuration page.
2. Click **Update** next to **VPC Whitelist**.
3. In the **Edit VPC Whitelist** pane, enter IP addresses.

   You can enter both IP addresses and CIDR blocks. For example, enter `192.168.0.1` or `192.168.0.0/24`. Separate multiple IP addresses or CIDR blocks with commas (,). You can enter `127.0.0.1` to deny requests from all IPv4 addresses or enter `0.0.0.0/0` to allow requests from all IPv4 addresses.

   > Note
   > - By default, requests from all IPv4 addresses within the VPC in which the Elasticsearch cluster resides are allowed.
   > - The VPC whitelist is used to control access from IP addresses within VPCs.

4. Click **OK**.

10.3.9. Back up data

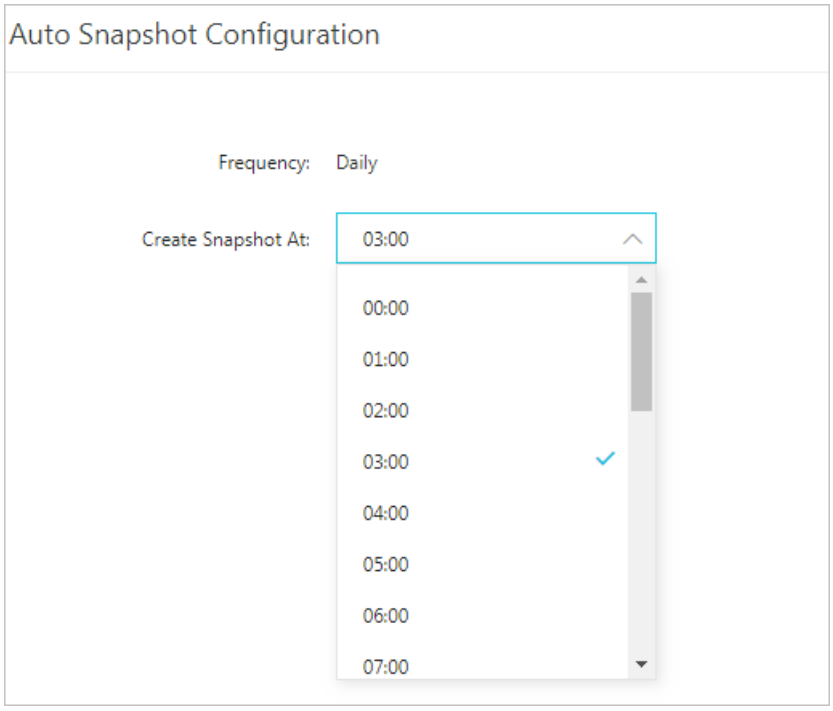
10.3.9.1. Enable and configure the auto snapshot feature

The auto snapshot feature allows you to specify the snapshot creation period and time. After you specify them, the system automatically creates snapshots. This topic describes how to enable and configure the auto snapshot feature.

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Snapshots**.
4. Turn on the **Auto Snapshot** switch. This switch is turned on by default.

   > Notice After the auto snapshot feature is enabled, the system uses the system time of the region where your Elasticsearch cluster resides to create snapshots. Do not perform snapshot operations when the system is creating snapshots.

5. Click **Modify Configuration** on the right side of **Snapshots (Free Trial)**.
6. In the **Auto Snapshot Configuration** pane, set **Create Snapshot At**.



- **Frequency**: The value of this parameter is Daily.
- **Create Snapshot At**: specifies the hour for creating a snapshot every day. Valid values: 00:00 to 23:00.

7. Click **Save**.

10.3.9.2. Query snapshot status

After you enable the auto snapshot feature, you can call the snapshot operation to query the status of snapshots. This topic describes how to query the status of snapshots.

> **Note** You can run all the commands provided in this topic in the Kibana console. For information about how to log on to the Kibana console, see Access an Elasticsearch cluster.

Query all snapshots

Run the following command to query information about all snapshots that are stored in the aliyun_auto_snapshot repository:

```
GET _snapshot/aliyun_auto_snapshot/_all
```

If the command is successfully executed, the following result is returned:

```
{
 "snapshots": [
  {
   "snapshot": "<yourSnapshotName>",
   "uuid": "n7YIayyZTm2hwg8BeW****",
   "version_id": 5050399,
```

```
    "version": "5.5.3",
    "indices": [
      ".kibana"
    ],
    "state": "SUCCESS",
    "start_time": "2018-06-28T01:22:39.609Z",
    "start_time_in_millis": 1530148959609,
    "end_time": "2018-06-28T01:22:39.923Z",
    "end_time_in_millis": 1530148959923,
    "duration_in_millis": 314,
    "failures": [],
    "shards": {
      "total": 1,
      "failed": 0,
      "successful": 1
    }
  },
  {
    "snapshot": "<yourSnapshotName>",
    "uuid": "frdl1YFzQ5Cn5xN9ZW****",
    "version_id": 5050399,
    "version": "5.5.3",
    "indices": [
      ".kibana"
    ],
    "state": "SUCCESS",
    "start_time": "2018-06-28T01:25:00.764Z",
    "start_time_in_millis": 1530149100764,
    "end_time": "2018-06-28T01:25:01.482Z",
    "end_time_in_millis": 1530149101482,
    "duration_in_millis": 718,
    "failures": [],
    "shards": {
      "total": 1,
      "failed": 0,
      "successful": 1
    }
  }
 ]
}
```

The state field indicates the status of a snapshot. A snapshot can be in one of the following states:

- `IN_PROGRESS` : The snapshot is being created.
- `SUCCESS` : The snapshot is created, and all shards are stored.
- `FAILED` : The snapshot fails to be created because some shards cannot be stored.
- `PARTIAL` : The snapshot is created, but a minimum of one shard fails to be stored.
- `INCOMPATIBLE` : The snapshot is incompatible with the Elasticsearch cluster version.

Query a specified snapshot

Run the following command to query information about a specified snapshot that is stored in the aliyun_auto_snapshot repository:

```
GET _snapshot/aliyun_auto_snapshot/<snapshot>/_status
```

Replace `<snapshot>` with the name of the snapshot, such as `<yourSnapshotName>` .

If the command is successfully executed, the following result is returned:

```
{
 "snapshots": [
  {
   "snapshot": "<yourSnapshotName>",
   "repository": "aliyun_auto_snapshot",
   "uuid": "n7YIayyZTm2hwg8BeWbydA",
   "state": "SUCCESS",
   "shards_stats": {
    "initializing": 0,
    "started": 0,
    "finalizing": 0,
    "done": 1,
    "failed": 0,
    "total": 1
   },
   "stats": {
    "number_of_files": 4,
    "processed_files": 4,
    "total_size_in_bytes": 3296,
    "processed_size_in_bytes": 3296,
    "start_time_in_millis": 1530148959688,
    "time_in_millis": 77
   },
   "indices": {
```

```
    ".kibana": {
     "shards_stats": {
      "initializing": 0,
      "started": 0,
      "finalizing": 0,
      "done": 1,
      "failed": 0,
      "total": 1
     },
     "stats": {
      "number_of_files": 4,
      "processed_files": 4,
      "total_size_in_bytes": 3296,
      "processed_size_in_bytes": 3296,
      "start_time_in_millis": 1530148959688,
      "time_in_millis": 77
     },
     "shards": {
      "0": {
       "stage": "DONE",
       "stats": {
        "number_of_files": 4,
        "processed_files": 4,
        "total_size_in_bytes": 3296,
        "processed_size_in_bytes": 3296,
        "start_time_in_millis": 1530148959688,
        "time_in_millis": 77
       }
      }
     }
    }
   }
 ]
}
```


10.3.9.3. Restore data from automatic snapshots

If you enable the auto snapshot feature for an Elasticsearch cluster, the system creates snapshots for the cluster every day. You can call the restore operation to restore data to the cluster from the created snapshots. This topic describes how to restore data from automatic snapshots.

> **Note** You can run all the commands provided in this topic in the Kibana console. For information about how to log on to the Kibana console, see Access an Elasticsearch cluster.

Background information

- The first snapshot is a full copy of the data in an Elasticsearch cluster. Subsequent snapshots store only incremental data. Therefore, it requires longer time to create the first snapshot than a subsequent snapshot.
- The system stores automatic snapshots that are created only within the last five days.
- A snapshot does not store monitoring data generated by an Elasticsearch cluster, such as indexes with the prefix `.monitoring` or `.security_audit`.
- An automatic snapshot repository is created when the first snapshot is created.

Query all snapshot repositories

Run the following command to query all snapshot repositories:

```
GET _snapshot
```

If the command is successfully executed, the following result is returned:

```
{
 "aliyun_auto_snapshot": {
  "type": "oss",
  "settings": {
   "compress": "true",
   "base_path": "xxxx",
   "endpoint": "xxxx"
  }
 }
}
```

- aliyun_auto_snapshot: the name of the repository.
- type: the storage where snapshots are stored.
- compress: indicates whether compression is enabled. The value true indicates that the metadata of indexes is compressed during snapshot creation.
- base_path: the location of snapshots in Object Storage Service (OSS).
- endpoint: the endpoint of the OSS bucket.

The auto snapshot feature also supports the following default parameters that are not displayed:

- `max_snapshot_bytes_per_sec:40mb` : The maximum speed for snapshot creation on a single node is 40 MB/s.
- `max_restore_bytes_per_sec:40mb` : The maximum speed for data restoration on a single node is 40 MB/s.
- `chunk_size: Max 1Gb` : During snapshot creation, a large index is divided into multiple parts. The maximum size of each part is 1 GB.

Query all snapshots

Run the following command to query information about all snapshots that are stored in the aliyun_auto_snapshot repository:

```
GET _snapshot/aliyun_auto_snapshot/_all
```

If the command is successfully executed, the following result is returned:

```
{
 "snapshots": [
  {
   "snapshot": "<yourSnapshotName>",
   "uuid": "MMRniVLPRAiawSCm8D****",
   "version_id": 5050399,
   "version": "5.5.3",
   "indices": [
    "index_1",
    ".security",
    ".kibana"
   ],
   "state": "SUCCESS",
   "start_time": "2018-06-27T01:16:01.009Z",
   "start_time_in_millis": 1530062161009,
   "end_time": "2018-06-27T01:16:05.632Z",
   "end_time_in_millis": 1530062165632,
   "duration_in_millis": 4623,
   "failures": [],
   "shards": {
    "total": 12,
    "failed": 0,
    "successful": 12
   }
  }
 ]
}
```


Restore indexes from a snapshot

You can call the `_restore` operation to restore indexes from snapshots.

- Run the following command to restore all indexes from a specified snapshot that is stored in the aliyun_auto_snapshot repository. The restoration task is executed at the backend.

  ```
  POST _snapshot/aliyun_auto_snapshot/<snapshot>/_restore
  ```

  Replace `<snapshot>` with the name of the snapshot, such as `<yourSnapshotName>`.

- Run the following command to restore all indexes from a specified snapshot that is stored in the aliyun_auto_snapshot repository. Then, wait until the restoration task is complete.

The `_restore` operation runs restoration tasks asynchronously. An Elasticsearch cluster immediately returns a response if the _restore operation can be executed. The restoration task is executed at the backend. You can specify the `wait_for_completion` parameter to enable the cluster to return a response only after the restoration task is complete.

```
POST _snapshot/aliyun_auto_snapshot/<snapshot>/_restore?wait_for_completion=true
```

Replace `<snapshot>` with the name of the snapshot, such as `<yourSnapshotName>`.

- Run the following command to restore specified indexes from a specific snapshot that is stored in the aliyun_auto_snapshot repository, and rename the restored indexes. The restoration task is executed at the backend.

```
POST _snapshot/aliyun_auto_snapshot/<snapshot>/_restore
{
"indices": "index_1",
"rename_pattern": "index_(.+)",
"rename_replacement": "restored_index_$1"
}
```

  - <snapshot>: Replace it with the name of the snapshot, such as `<yourSnapshotName>`.
  - indices: the name of the index you want to restore.
  - rename_pattern: optional. This parameter specifies the regular expression that is used to match the name of the index you want to restore.
  - rename_replacement: optional. This parameter specifies the regular expression that is used to rename a matched index.

10.3.9.4. Commands for creating snapshots and restoring data

You can call the snapshot operation to back up or restore data for your Apsara Stack Elasticsearch cluster. The snapshot operation retrieves the status and data of your cluster and then stores them to a shared repository.

The first snapshot is a full copy of the data in a cluster. Subsequent snapshots store only incremental data. Therefore, when you create subsequent snapshots, the system only needs to add data to or remove data from the snapshots. This means that it requires less time to create a subsequent snapshot than the first snapshot.

Precautions

- This topic uses the following markers to provide descriptions for code: <1>, <2>, and <3>. Remove these markers when you run the code.
- You can run the code provided in this topic in the Kibana console of your Elasticsearch cluster. For more information, see Access an Elasticsearch cluster.

Create a repository

```
PUT _snapshot/my_backup
{
  "type": "oss",
   "settings": {
      "endpoint": "http://oss-cn-hangzhou-internal.aliyuncs.com", <1>
      "access_key_id": "xxxx",
      "secret_access_key": "xxxxxx",
      "bucket": "xxxxxx", <2>
      "compress": true,
      "base_path": "snapshot/" <3>
   }
}
```

- <1>: The `endpoint` parameter specifies the internal endpoint of the OSS bucket.
- <2>: The bucket parameter specifies the name of an OSS bucket that has been created.
- <3>: The base_path parameter specifies the path of the repository. The default value is the root directory.

Set the size of each part

When you upload large volumes of data to an OSS bucket, you can use the `chunk_size` parameter to set the size of each part. This allows you to upload the data in multiple parts.

```
POST _snapshot/my_backup/ <1>
{
   "type": "oss",
   "settings": {
      "endpoint": "http://oss-cn-hangzhou-internal.aliyuncs.com",
      "access_key_id": "xxxx",
      "secret_access_key": "xxxxxx",
      "bucket": "xxxxxx",
      "chunk_size": "500mb",
      "base_path": "snapshot/" <2>
   }
}
```

- <1>: Use the POST method instead of the PUT method. The POST method updates repository settings.
- <2>: The base_path parameter specifies the path of the repository. The default value is the root directory.

Query repository information

```
GET _snapshot
```

You can also use the `GET _snapshot/my_backup` command to query the information of a specified repository.

Create snapshots

The following command is a basic command that is used to create a snapshot:

```
PUT _snapshot/my_backup/snapshot_1
```

This command creates the `snapshot_1` snapshot for all open indexes. The snapshot is stored in the `my_backup` repository. After you run the command, the system immediately returns a response while the snapshot is created at the backend.

If you want the system to return a response after it creates the snapshot, add the `wait_for_completion` parameter to the command.

```
PUT _snapshot/my_backup/snapshot_1? wait_for_completion=true
```

After you run the command, the system does not return a response until the snapshot is created. If the size of the index is large, the response is returned after a longer period of time.

Create a snapshot for specified indexes

By default, a snapshot contains all open indexes. For Kibana, when you create a snapshot, you may want to ignore all diagnostic indexes (the `.kibana` indexes) because of limited disk space. To create a snapshot for specified indexes, run the following command:

> **Notice** A repository stores multiple snapshots. Each snapshot is a copy of all indexes, specified indexes, or a single index in a cluster. When you create a snapshot, make sure that the snapshot name is unique.

```
PUT _snapshot/my_backup/snapshot_2
{
  "indices": "index_1,index_2"
}
```

The preceding command creates a snapshot only for the `index1` and `index2` indexes.

Query snapshot information

In some cases, you may need to query snapshot information. For example, a snapshot name containing a date is hard to remember, such as `backup_2014_10_28`.

To query the information of a snapshot, send a `GET` request that contains both the repository name and snapshot name. Example:

```
GET _snapshot/my_backup/snapshot_2
```

The following response contains detailed information of the snapshot:

```
{
"snapshots": [
  {
    "snapshot": "snapshot_2",
    "indices": [
      ".marvel_2014_28_10",
      "index1",
      "index2"
    ],
    "state": "SUCCESS",
    "start_time": "2014-09-02T13:01:43.115Z",
    "start_time_in_millis": 1409662903115,
    "end_time": "2014-09-02T13:01:43.439Z",
    "end_time_in_millis": 1409662903439,
    "duration_in_millis": 324,
    "failures": [],
    "shards": {
      "total": 10,
      "failed": 0,
      "successful": 10
    }
  }
]
}
```

You can replace the snapshot name in the preceding command with `_all` to query all snapshots in the repository. Example:

```
GET _snapshot/my_backup/_all
```

Delete a snapshot

You can specify a repository name and a snapshot name, and send a `DELETE` request to delete the specified snapshot. Example:

```
DELETE _snapshot/my_backup/snapshot_2
```

> **Notice**
>
> - You can only use the DELETE operation to delete snapshots. A snapshot is associated with other backup files. Some of the files may also be used by other snapshots. The DELETE operation does not delete files that are still being used by other snapshots. It only deletes files that are associated with the deleted snapshot and are not used by other snapshots.
> - If you choose to manually delete a snapshot, you may delete files that are associated with snapshots by mistake. This may cause data loss.

Monitor snapshot creation progress

The `wait_for_completion` parameter provides a simple method for you to monitor the progress of a snapshot creation task. However, this parameter is not suitable for snapshot creation tasks of medium-size Elasticsearch clusters. You can use one of the following methods to query detailed information about a snapshot:

- Send a `GET` request with the snapshot name specified. Example:

  ```
  GET _snapshot/my_backup/snapshot_3
  ```

  If the system is still creating the snapshot when you run the preceding command, the information of the creation task is returned, such as the time when the snapshot creation task started and the duration.

  > **Notice** The preceding command shares a thread pool with the command used to create a snapshot. Therefore, if you create a snapshot for large shards, the preceding command has to wait until the resources that are used by the snapshot creation command in the thread pool are released.

- Call the _status operation to query the snapshot status.

  ```
  {
  "snapshots": [
    {
      "snapshot": "snapshot_3",
      "repository": "my_backup",
      "state": "IN_PROGRESS", <1>
      "shards_stats": {
        "initializing": 0,
        "started": 1, <2>
        "finalizing": 0,
        "done": 4,
        "failed": 0,
        "total": 5
      },
      "stats": {
  ```

```
    "number_of_files": 5,
    "processed_files": 5,
    "total_size_in_bytes": 1792,
    "processed_size_in_bytes": 1792,
    "start_time_in_millis": 1409663054859,
    "time_in_millis": 64
  },
  "indices": {
    "index_3": {
      "shards_stats": {
        "initializing": 0,
        "started": 0,
        "finalizing": 0,
        "done": 5,
        "failed": 0,
        "total": 5
      },
      "stats": {
        "number_of_files": 5,
        "processed_files": 5,
        "total_size_in_bytes": 1792,
        "processed_size_in_bytes": 1792,
        "start_time_in_millis": 1409663054859,
        "time_in_millis": 64
      },
      "shards": {
        "0": {
          "stage": "DONE",
          "stats": {
            "number_of_files": 1,
            "processed_files": 1,
            "total_size_in_bytes": 514,
            "processed_size_in_bytes": 514,
            "start_time_in_millis": 1409663054862,
            "time_in_millis": 22
          }
        },
        ...
```

- <1>: The status of the snapshot. If a snapshot is being created, the value of the field is `IN_PROGRESS`.
- <2>: The number of shards that are being transmitted. If the value 1 is returned, a shard of the snapshot is being transmitted, and the other four shards have been transmitted.

  The value of the `shards_stats` parameter contains the status of the snapshot. It also contains statistics about each index and shard. This parameter allows you to learn the detailed information of the snapshot creation progress. A shard can be in one of the following states:

  - `INITIALIZING`: The shard is verifying the status of the cluster to check whether the shard can be stored in a snapshot. In most cases, this process is fast.
  - `STARTED`: Data is being transmitted to the repository.
  - `FINALIZING`: The data transmission process is complete. The shard is sending snapshot metadata.
  - `DONE`: The snapshot is created.
  - `FAILED`: An error occurred during the snapshot creation. The shard, index, or snapshot cannot be processed. You can view logs for more information.

Cancel a snapshot

To cancel a snapshot, run the following command when the snapshot is being created:

```
DELETE _snapshot/my_backup/snapshot_3
```

This command stops the snapshot creation process and deletes the snapshot that is being created from the repository.

Restore indexes from a snapshot

To restore indexes from a snapshot, run the command that is used in the "Create a repository" section on the Elasticsearch cluster that stores these indexes. You can use one of the following methods to restore indexes from a snapshot:

- To restore indexes from a specified snapshot, append the `_restore` parameter to the snapshot name in the command to run. Example:

  ```
  POST _snapshot/my_backup/snapshot_1/_restore
  ```

  After you run this command, the system restores all indexes in the snapshot. For example, if the `snapshot_1` snapshot contains five indexes, all these indexes are restored to the Elasticsearch cluster. You can also specify the indexes that you want to restore. For more information, see Create a snapshot for specified indexes.
- Restore specified indexes and rename the indexes. If you only want to verify or process the data in indexes and do not need to overwrite the data, use this method to restore the indexes.

```
POST /_snapshot/my_backup/snapshot_1/_restore
{
 "indices": "index_1", <1>
 "rename_pattern": "index_(.+)", <2>
 "rename_replacement": "restored_index_$1" <3>
}
```

In this example, the `index_1` index is restored to your Elasticsearch cluster and renamed `restored_index_1`.

- <1>: The system only restores the `index_1` index from the snapshot.
- <2>: The system searches for the index that is being restored and matches the index name with the provided pattern.
- <3>: The system renames the matching index.

- If you want the system to return a response after it restores the index, add the `wait_for_completion` parameter to the command.

```
POST _snapshot/my_backup/snapshot_1/_restore? wait_for_completion=true
```

After you call the _restore operation, the system immediately returns a response and restores the index at the backend. If you want the operation to return the result after the restore process is complete, add the `wait_for_completion` parameter.

Monitor index restoration progress

> **Note** Restoring data from a repository applies the existing restoration mechanism in Elasticsearch. Restoring shards from a repository is the same as restoring data from a node.

You can call the recovery operation to monitor the progress of an index restoration task.

- Monitor a specified index that is being restored.

```
GET restored_index_3/_recovery
```

The recovery operation is a general-purpose operation. It shows the status of the shards that are being transmitted to your cluster.

- Monitor all indexes on the cluster. This may include shards that are irrelevant to the restoration process.

```
GET /_recovery/
```

The following content shows the sample response:

```
{
"restored_index_3" : {
 "shards" : [ {
  "id" : 0,
```

```
  "type" : "snapshot", <1>
  "stage" : "index",
  "primary" : true,
  "start_time" : "2014-02-24T12:15:59.716",
  "stop_time" : 0,
  "total_time_in_millis" : 175576,
  "source" : { <2>
    "repository" : "my_backup",
    "snapshot" : "snapshot_3",
    "index" : "restored_index_3"
  },
  "target" : {
    "id" : "ryqJ5lO5S4-lSFbGntkEkg",
    "hostname" : "my.fqdn",
    "ip" : "10.0. **.**",
    "name" : "my_es_node"
  },
  "index" : {
    "files" : {
      "total" : 73,
      "reused" : 0,
      "recovered" : 69,
      "percent" : "94.5%" <3>
    },
    "bytes" : {
      "total" : 79063092,
      "reused" : 0,
      "recovered" : 68891939,
      "percent" : "87.1%"
    },
    "total_time_in_millis" : 0
  },
  "translog" : {
    "recovered" : 0,
    "total_time_in_millis" : 0
  },
  "start" : {
    "check_index_time" : 0,
    "total_time_in_millis" : 0
  }
}]
```

```
}
}
```

- <1>: The `type` parameter indicates the type of restoration. The value `snapshot` indicates that the shard is being restored from a snapshot.
- <2>: The `source` parameter indicates the source snapshot and repository.
- <3>: The `percent` parameter indicates the progress of the restoration task. The value `94.5%` indicates that 94.5% of the shard files are restored.

The response lists all indexes that are being restored and the shards in these indexes. Each shard has statistics about the start or end time, duration, restoration progress, and bytes transmitted.

Cancel index restoration

To cancel index restoration, you only need to delete the indexes that are being restored. A restoration process is a shard restoration process. You can call the DELETE operation to modify the status of the cluster and cancel the restoration process. Example:

```
DELETE /restored_index_3
```

If you run the preceding command when the `restored_index_3` index is being restored, the system stops the restoration and deletes the data that has been restored to the cluster. For more information, see Snapshot And Restore.

Use a snapshot to migrate data

To use a snapshot to migrate data from an Elasticsearch cluster to another, follow these steps:

1. Back up a snapshot to OSS.
2. Create a snapshot repository on the destination cluster. The repository must use the OSS bucket that stores the snapshot.
3. Set the `base_path` parameter to the path of the snapshot.
4. Run the data restoration command on the destination cluster.

> **Note** These steps provide a simple solution to snapshot-based data migration. For more information, see Use a snapshot stored in OSS to migrate Elasticsearch data.

10.4. Manage documents

10.4.1. Create a document

This topic describes how to use Elasticsearch to create a document.

Access an Elasticsearch cluster and send a POST request to create a document in the cluster.

```
curl -XPOST http://<HOST>:<PORT>/my_index/my_type -d '{"title": "One", "tags": ["ruby"]}'
```

- `<HOST>`: the internal endpoint of the Elasticsearch cluster.
- `<PORT>`: the internal port of the Elasticsearch cluster. The default port is 9200.
- `my_index`: the name of the index, which can be customized.
- `my_type`: the type of the document, which can be customized.

> **Note**
>
> - Each document has an ID and a type. The response contains the ID and type of the document. If you do not specify an ID or type when you create a document, the system automatically generates one for the document.
> - If you enable the Auto Indexing feature and the specified index name does not exist, the system automatically creates an index when creating the document. The Auto Indexing feature is disabled by default. For information about how to enable this feature, see Configure a YML file.

If the command is successfully executed, the following result is returned:

```
{
  "_index": "my_index",
  "_type": "my_type",
  "_id": "AV4JIvi15ny3i8DCdK1H",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

10.4.2. Update a document

This topic describes how to use Elasticsearch to update a document.

Access an Elasticsearch cluster and send a POST request to update a document in the cluster.

```
curl -XPOST http://<HOST>:<PORT>/my_index/my_type/<doc_id>
```

- `<HOST>`: the internal endpoint of the Elasticsearch cluster.
- `<PORT>`: the internal port of the Elasticsearch cluster. The default port is `9200`.
- `my_index`: the name of the index.

- `my_type` : the type of the document.
- `<doc_id>` : the ID of the document.

Example:

```
curl -XPOST http://<HOST>:<PORT>/my_index/my_type/AV4JIVI15NY3I8DCDK1H
-d '{"title": "FOUR UPDATED", "TAGS": ["RUBY", "PHP"]}'
```

If the command is successfully executed, the following result is returned:

```
{
   "_index": "my_index",
   "_type": "my_type",
   "_id": "AV4JIvi15ny3i8DCdK1H",
   "_version": 2,
   "result": "updated",
   "_shards": {
      "total": 2,
      "successful": 1,
      "failed": 0
   },
   "created": false
}
```

10.4.3. Retrieve a document

This topic describes how to use Elasticsearch to retrieve a document.

Access an Elasticsearch cluster and send a GET request to retrieve a document in the cluster.

Example:

```
curl -XGET http://<HOST>:<PORT>/my_index/my_type/AV4JIVI15NY3I8DCDK1H
```

- `<HOST>` : the internal endpoint of the Elasticsearch cluster.
- `<PORT>` : the internal port of the Elasticsearch cluster. The default port is `9200` .
- `my_index` : the name of the index.
- `my_type` : the type of the document.
- `AV4JIVI15NY3I8DCDK1H` : the ID of the document.

If the command is successfully executed, the following result is returned:

```
{
  "_INDEX" : "MY_INDEX",
  "_TYPE" : "MY_TYPE",
  "_ID" : "AV4JIVI15NY3I8DCDK1H",
  "_VERSION" : 2,
  "EXISTS" : TRUE,
  "_SOURCE" : {
  "TITLE": "FOUR UPDATED", "TAGS": ["RUBY", "PHP"]
  }
```

10.4.4. Search for documents

This topic describes how to use Elasticsearch to search for documents.

Access an Elasticsearch cluster and send a GET or POST request to search for documents in the cluster. You can specify URI parameters in the request. Examples:

```
curl -XGET http://<HOST>:<PORT>/_search
curl -XGET http://<HOST>:<PORT>/{index_name}/_search
curl -XGET http://<HOST>:<PORT>/{index_name}/{type_name}/_search
```

To search for documents in which the title field contains the T keyword, run the following command:

```
curl -XGET http://<HOST>:<PORT>/my_index/my_type/_search?q=title:T*
```

- `<HOST>` : the internal endpoint of the Elasticsearch cluster.
- `<PORT>` : the internal port of the Elasticsearch cluster. The default port is `9200` .
- `my_index` : the name of the index.
- `my_type` : the type of the document.

10.4.5. Perform a complex search

This topic describes how to use Elasticsearch to perform a complex search.

Access an Elasticsearch cluster and send a POST request to perform a complex search on documents in the cluster. Example:

```
$ curl -XPOST http://<HOST>:<PORT>/my_index/my_type/_search?pretty=true -d '{
"query": {
   "query_string": {"query": "*"}
},
"facets": {
   "tags": {
      "terms": {"field": "tags"}
   }
}}'
```

- `<HOST>`: the internal endpoint of the Elasticsearch cluster.
- `<PORT>`: the internal port of the Elasticsearch cluster. The default port is `9200`.
- `my_index`: the name of the index.
- `my_type`: the type of the document.

> **Note** The `?pretty=true` parameter is used to make the result easier to understand.

10.4.6. Delete a document

This topic describes how to use Elasticsearch to delete a document.

Access an Elasticsearch cluster and call the DELETE operation to delete a document in the cluster.

Delete a document with its ID specified

```
curl -XDELETE http://<HOST>:<PORT>/my_index/my_type/{ID}
```

- `<HOST>`: the internal endpoint of the Elasticsearch cluster.
- `<PORT>`: the internal port of the Elasticsearch cluster. The default port is `9200`.
- `my_index`: the name of the index.
- `my_type`: the type of the document.
- `ID`: the ID of the document.

Delete a specified type of documents

```
curl -XDELETE http://<HOST>:<PORT>/my_index/my_type/
```

- `<HOST>`: the internal network endpoint of the Elasticsearch cluster.
- `<PORT>`: the internal port of the Elasticsearch cluster. The default port is `9200`.

- `my_index` : the name of the index.
- `my_type` : the type of the documents.

Delete documents in a specified index

```
curl -XDELETE http://<HOST>:<PORT>/{my_index}
```

- `<HOST>` : the internal endpoint of the Elasticsearch cluster.
- `<PORT>` : the internal port of the Elasticsearch cluster. The default port is `9200` .
- `my_index` : the name of the index.

10.5. Elasticsearch test

After you create an Elasticsearch instance, you can log on to the Kibana console integrated into the Elasticsearch console and test the search function on the Dev Tools page. You can also run the curl command in an ECS instance that meets the requirements to perform the test.

10.5.1. Use a curl command to access an Elasticsearch cluster over port 9200

This topic describes how to use a curl command to access an Elasticsearch cluster over port 9200.

The following example shows how to access an Elasticsearch cluster, with its username and password specified:

```
curl -u username:password 'http://<HOST>:9200/filebeat/my_type/'?pretty -d '{"title": "One", "tags": [
"ruby"]}'
```

If the command is successfully executed, the following result is returned:

```
{
 "_index" : "filebeat",
 "_type" : "my_type",
 "_id" : "AV-bTkaTwdiHxfaSqlAt",
 "_version" : 1,
 "result" : "created",
 "_shards" : {
  "total" : 2,
  "successful" : 2,
  "failed" : 0
 },
 "created" : true
}
```

The following example shows how to access an Elasticsearch cluster, with its username and password not specified:

```
curl http://<HOST>:9200/index_name/type_name -XPOST -d '{"title": "One", "tags": ["ruby"]}'
```

10.5.2. Use Python to access an Elasticsearch cluster over port 9200

This topic describes how to use Python to access an Elasticsearch cluster over port 9200.

```
from elasticsearch import Elasticsearch, RequestsHttpConnection
import certifi
es = Elasticsearch(
    ['<HOST>'],
    http_auth=('username', 'password'),
    port=9200,
    use_ssl=False
)
res = es.index(index="my_index", doc_type="my_type", id=1, body={"title": "One", "tags": ["ruby"]})
print(res['created'])
res = es.get(index="my-index", doc_type="my-type", id=1)
print(res['_source'])
```

10.5.3. Use Java REST Client to access an Elasticsearch cluster over port 9200

This topic describes how to use Java REST Client to access an Elasticsearch cluster over port 9200 and provides related precautions.

Precautions

- The open source Elasticsearch team no longer maintains Transport Client. We recommend that you do not use Transport Client to access an Elasticsearch cluster. If you use Transport Client 5.5.3 to access an Elasticsearch cluster, the system displays the "NoNodeAvailableException" error message. We recommend that you use Java Low Level REST Client to access an Elasticsearch cluster.
- Java REST Client described in this topic is applicable only to Elasticsearch V5.5.3 clusters. For more information about how to use Java REST Client to access an Elasticsearch V6.3.2 cluster, see Java REST Client 6.3.2.
- The version of Java REST Client must be the same as that of your Elasticsearch cluster.

Prerequisites

- An Elasticsearch cluster is created, and the Auto Indexing feature is enabled for the cluster.

  For more information, see Create an Elasticsearch cluster and Configure a YML file.
- The relevant JDK is installed, and environment variables are configured. The JDK version must be 1.8 or later.

Sample code

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.entity.ContentType;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;
import org.apache.http.nio.entity.NStringEntity;
import org.apache.http.util.EntityUtils;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import java.io.IOException;
import java.util.Collections;
public class RestClientTest {
   public  static void main(String[]args){
      final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
```

```
        credentialsProvider.setCredentials(AuthScope.ANY,
                new UsernamePasswordCredentials("<USER NAME>", "<PASSWORD>"));
        RestClient restClient = RestClient.builder(new HttpHost("<HOST>", 9200))
                .setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {
                    @Override
                    public HttpAsyncClientBuilder customizeHttpClient(HttpAsyncClientBuilder httpClientBuilder
) {
                        return httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
                    }
                }).build();
        try {
            //index a document
            HttpEntity entity = new NStringEntity("{\n\"user\" : \"kimchy\"\n}", ContentType.APPLICATION_JS
ON);
            Response indexResponse = restClient.performRequest(
                    "PUT",
                    "/index/type/123",
                    Collections. <String, String>emptyMap(),
                    entity);
            //search a document
            Response response = restClient.performRequest("GET", "/index/type/123",
                    Collections.singletonMap("pretty", "true"));
            System.out.println(EntityUtils.toString(response.getEntity()));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- `<USER NAME>` : the username of your Elasticsearch cluster.
- `<PASSWORD>` : the password of your Elasticsearch cluster.
- `<HOST>` : the internal endpoint of your Elasticsearch cluster. You can obtain the endpoint from the Basic Information page of the cluster.

11.Elasticsearch (on k8s)

11.1. What is Apsara Stack Elasticsearch?

Open source Elasticsearch is a Lucene-based, distributed, real-time search and analytics engine. It is a product released under the Apache License. Elasticsearch is a popular search engine for enterprises. It provides distributed services, allowing you to store, query, and analyze large amounts of datasets in near real time. Elasticsearch is typically used as a basic engine or technology to support complex queries and high-performance applications.

Apsara Stack Elasticsearch provides fully-managed Elasticsearch services. It supports multiple versions of open source Elasticsearch and is compatible with all open source Elasticsearch features. Apsara Stack Elasticsearch offers an optimized kernel and provides the multi-tenancy, high availability, and auto scaling features. In addition to the features of open source Elasticsearch, Apsara Stack Elasticsearch allows you to create a cluster in a visualized manner, use Migration Assistant to migrate data, manage repositories, create snapshots, manage plug-ins, and perform O&M operations.

11.2. Quick start

11.2.1. Log on to the Elasticsearch console

This topic describes how to log on to the Elasticsearch console.

Prerequisites

- The domain name of the ASCM console is obtained from the deployment personnel before you log on to the ASCM console.
- A browser is available. We recommend that you use the Google Chrome browser.

Procedure

1. In the address bar, enter the URL used to log on to the ASCM console. Press the Enter key.
2. Enter your username and password.

   Obtain the username and password used to log on to the console from the operations administrator.

   > **Note** When you log on to the ASCM console for the first time, you must change the password of your username. For security reasons, your password must meet the minimum complexity requirements. The password must be 8 to 20 characters in length and must contain at least two of the following character types:
   >
   > - Uppercase or lowercase letters.
   > - Digits.
   > - Special characters. Special characters include exclamation points (!), at signs (@), number signs (#), dollar signs ($), and percent signs (%).

3. Click **Login** to go to the ASCM console homepage.
4. In the top navigation bar, choose **Products** > **Big Data** > **Elasticsearch(k8s)**.

5. Specify **Organization** and **Region** and click **Elasticsearch(k8s)**.

11.2.2. Access an Elasticsearch cluster

You can access an Elasticsearch cluster from an Elastic Compute Service (ECS) instance or the Kibana console.

Prerequisites

- Operations and maintenance (O&M) personnel have created an Elasticsearch cluster under your organization.

  For more information, see Elasticsearch Operations and Maintenance Guide.
- An ECS instance is created. The ECS instance must reside in the same Virtual Private Cloud (VPC), Kubernetes cluster, region, and zone as the Elasticsearch cluster.

Use an ECS instance to access an Elasticsearch cluster

1. Log on to the ECS instance over SSH and install the cURL tool.

   > **Note** For more information about other logon methods, see ECS User Guide.

2. Run the following command to connect to the internal endpoint of the Elasticsearch cluster.

   ```
   curl http://<HOST>:<PORT>
   ```

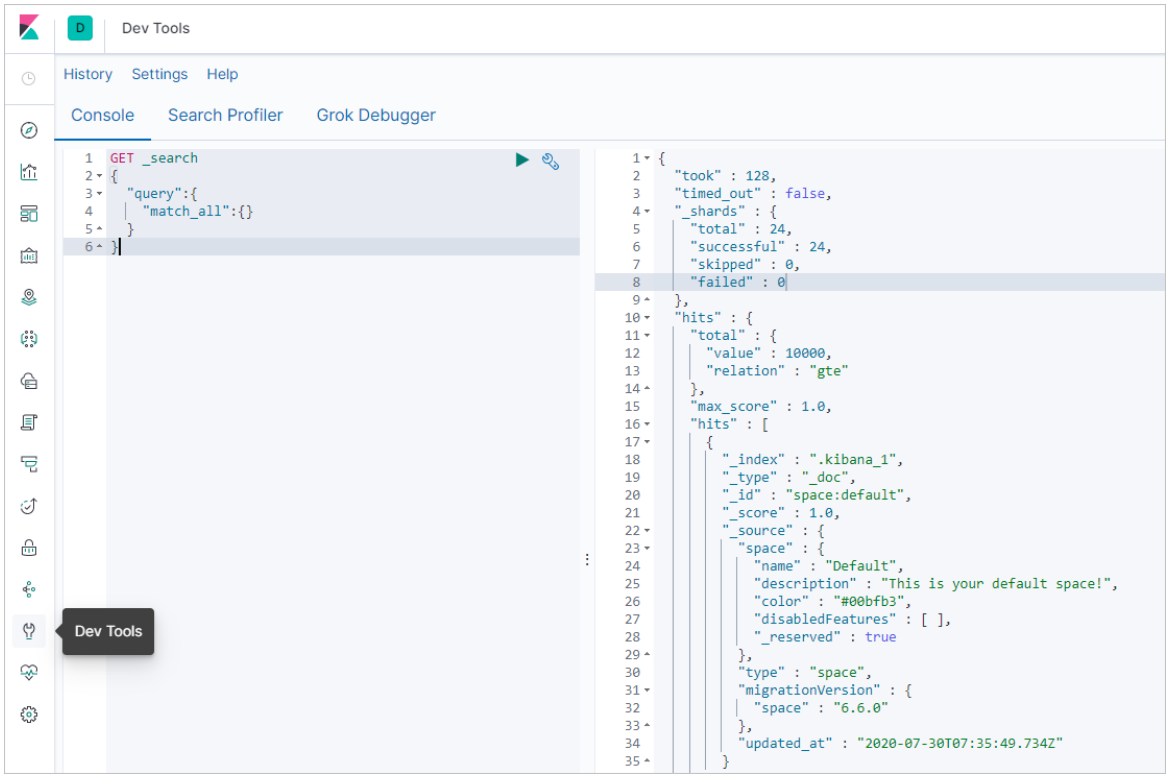
   - `<HOST>`: the internal endpoint of the Elasticsearch cluster. You can obtain the information from the Basic Information page of the cluster.

     > **Note** To go to the Basic Information page, log on to the Elasticsearch console, find the cluster, and click its ID in the Instance ID/Name column.

   - `<PORT>`: the internal port of the Elasticsearch cluster. You can obtain the information from the Basic Information page. The default port is 9200.

Use the Kibana console to access an Elasticsearch cluster

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Data Visualization**.
4. Click **Console** in the Kibana section.
5. On the page that appears, enter the username and password and click **Log in**.The username is elastic. You can obtain the password from O&M personnel.
6. In the left-side navigation pane, click the **Dev Tools** icon. On the **Console** tab, run the command shown in the following figure.

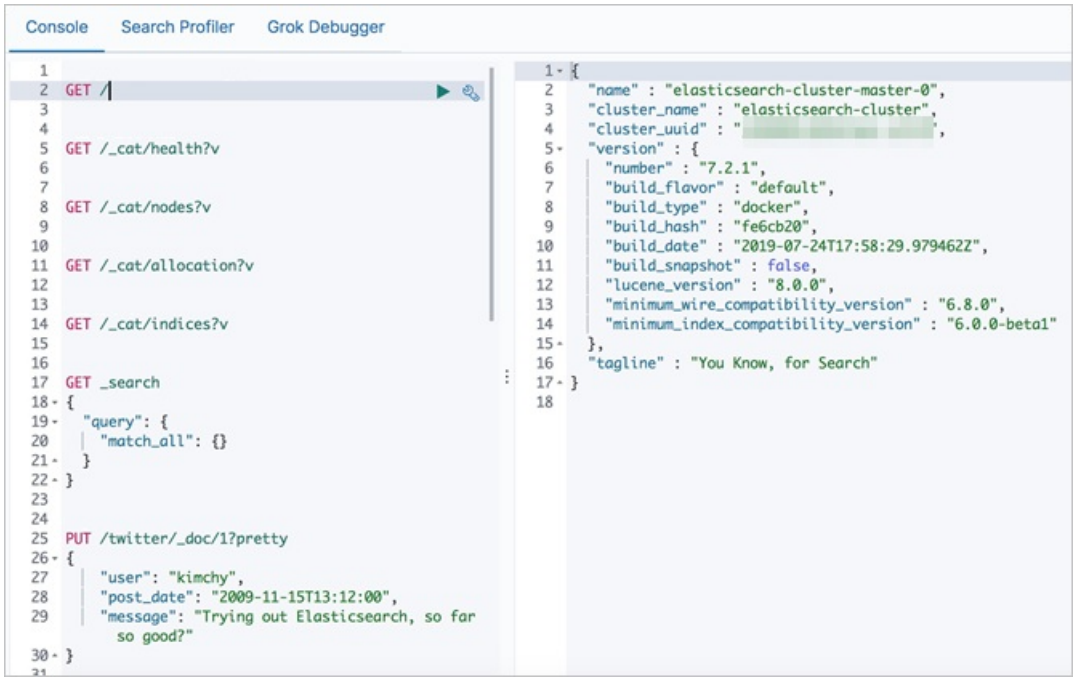


11.2.3. View the information of an Elasticsearch cluster

This topic describes how to view the information of an Elasticsearch cluster from the Kibana console.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following commands to view the basic information of the cluster.



- View the version of the cluster

  ```
  GET /
  ```

- View the health status of the cluster

  ```
  GET /_cat/health?v
  ```

- View the information of nodes in the cluster

  ```
  GET /_cat/nodes?v
  ```

- View JVM heap memory

  ```
  GET /_nodes/stats/jvm?pretty
  ```

- View information of disks

  ```
  GET /_cat/allocation?v
  ```

- View information of indexes

  ```
  GET /_cat/indices?v
  ```

11.2.4. Create an index

Before you use Elasticsearch to process documents, you must create an index. This topic describes how to create an index.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.

2. Run the following command to create an index:

```
PUT my_index?include_type_name=true
{
  "settings" : {
    "index" : {
      "number_of_shards" : "5",
      "number_of_replicas" : "1"
    }
  },
  "mappings" : {
    "_doc" : {
      "properties" : {
        "post_date": {
          "type": "date"
        },
        "tags": {
          "type": "keyword"
        },
        "title" : {
          "type" : "text",
          "analyzer" : "cjk"
        }
      }
    }
  }
}
```

Notice

- The preceding command is only for your reference. You can customize the parameters as needed. For more information, see open source Elasticsearch documentation.
- You can configure the YML configuration file of your Elasticsearch cluster to enable the Auto Indexing feature. After this feature is enabled, when you create a document, the system automatically creates an index for the document. Auto Indexing is used only for testing purposes. We recommend that you do not use this feature in production.
- Mapping types are deprecated in Elasticsearch V7.0.0 and later. Versions earlier than Elasticsearch V7.0.0 still support mapping types. For more information, see open source Elasticsearch documentation.

In the preceding example, an index named `my_index` is created, and the type of the index is `_doc`. The index is split into five shards, has one replica for each shard, and uses the `cjk` analyzer.

If the command is successfully executed, the following result is returned:

```
{
 "acknowledged" : true,
 "shards_acknowledged" : true,
 "index" : "my_index"
}
```

11.2.5. Manage documents

11.2.5.1. Create a document

This topic describes how to use Elasticsearch to create a document.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following command to create a document:

```
curl -XPUT http://<HOST>:<PORT>/<index_name>/_doc/<ID>
{
 "<field_name>": "<value>"
}'
```

   - `<HOST>`: the endpoint of the Elasticsearch cluster.
   - `<PORT>`: the port of the Elasticsearch cluster. The default port is 9200.
   - `<index_name>`: the name of the index.
   - `_doc`: the type of the index. The index type of Elasticsearch V7.0 and later must be `_doc`.
   - `<ID>`: the ID of the document.
   - `<field_name>`: the name of the field.
   - `value`: the value of the field.

   Examples:

```
curl -XPUT http://localhost:9200/my_index/_doc/1?pretty' -d '
{
 "title": "One",
 "tags": ["ruby"],
 "post_date":"2009-11-15T13:00:00"
}'
```

```
curl -XPUT http://localhost:9200/my_index/_doc/2?pretty' -d '
{
 "title": "Two",
 "tags": ["ruby"],
 "post_date":"2009-11-15T14:00:00"
}'
```

If the commands are successfully executed, the following results are returned:

```
{
 "_index" : "my_index",
 "_type" : "_doc",
 "_id" : "1",
 "_version" : 1,
 "result" : "created",
 "_shards" : {
  "total" : 2,
  "successful" : 2,
  "failed" : 0
 },
 "_seq_no" : 0,
 "_primary_term" : 1
}
```

```
{
 "_index" : "my_index",
 "_type" : "_doc",
 "_id" : "2",
 "_version" : 1,
 "result" : "created",
 "_shards" : {
  "total" : 2,
  "successful" : 2,
  "failed" : 0
 },
 "_seq_no" : 1,
 "_primary_term" : 1
}
```

For more information, see open source Elasticsearch documentation.

11.2.5.2. Update a document

This topic describes how to use Elasticsearch to update a document.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run one of the following commands to update a document:

   Update a whole document

   ```
   curl -XPUT http://<HOST>:<PORT>/<index_name>/_doc/<ID>
   {
    "<field_name>": "<value>"
   }'
   ```

   Make partial updates to a document

   ```
   curl -XPOST http://<HOST>:<PORT>/<index_name>/_update/<ID>
   {
    "script" : {
        "source": "ctx._source.<field_name> = '<value>'"
    }
   }'
   ```

   - `<HOST>` : the endpoint of the Elasticsearch cluster.
   - `<PORT>` : the port of the Elasticsearch cluster. The default port is `9200` .

- `<index_name>`: the name of the index.
- `_doc`: the type of the index.
- `<ID>`: the ID of the document.
- `<field_name>`: the name of the field.
- `value`: the value of the field.

Examples:

- Update a whole document

```
curl -XPUT http://localhost:9200/my_index/_doc/1?pretty' -d '
{
  "title": "One World One Dream",
  "tags": ["ruby"],
  "post_date":"2009-11-15T13:00:00"
}'
```

If the command is successfully executed, the following result is returned:

```
{
  "_index" : "my_index",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 2,
  "result" : "updated",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 1,
  "_primary_term" : 1
}
```

- Make partial updates to a document

```
curl -XPOST http://localhost:9200/my_index/_update/1?pretty' -d '
{
    "script" : {
        "source": "ctx._source.title='One World'"
    }
}'
```

If the command is successfully executed, the following result is returned:

```
{
  "_index" : "my_index",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 5,
  "result" : "updated",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 4,
  "_primary_term" : 1
}
```

You can also call the update operation to update documents. For more information, see open source Elasticsearch documentation.

11.2.5.3. Retrieve a document

This topic describes how to use Elasticsearch to retrieve a document.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following command to retrieve a document:

   ```
   curl -XGET http://<HOST>:<PORT>/<index_name>/_doc/<ID>
   ```

   - `<HOST>` : the endpoint of the Elasticsearch cluster.
   - `<PORT>` : the port of the Elasticsearch cluster.
   - `<index_name>` : the name of the index.
   - `_doc` : the type of the index.
   - `<ID>` : the ID of the document.

   Example:

   ```
   curl -XGET http://localhost:9200/my_index/_doc/1
   ```

   If the command is successfully executed, the following result is returned:

```
{
 "_index" : "my_index",
 "_type" : "_doc",
 "_id" : "1",
 "_version" : 5,
 "_seq_no" : 4,
 "_primary_term" : 1,
 "found" : true,
 "_source" : {
  "title" : "One World",
  "tags" : [
   "ruby",
   "blue"
  ],
  "post_date" : "2009-11-15T13:00:00"
 }
}
```

For more information, see open source Elasticsearch documentation.

11.2.5.4. Search for documents

This topic describes how to use Elasticsearch to search for documents.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following command to search for documents. URI parameters in the command are used to specify the object you want to search.

   ```
   curl -XGET http://<HOST>:<PORT>/_search
   curl -XGET http://<HOST>:<PORT>/<index_name>/_search
   ```

   - `<HOST>` : the endpoint of the Elasticsearch cluster.
   - `<PORT>` : the port of the Elasticsearch cluster. The default port is 9200.
   - `<index_name>` : the name of the index.

   To search for documents in which the title field contains the T keyword, run the following command:

   ```
   curl -XGET http://localhost:9200/my_index/_search?q=title:T*
   ```

   If the command is successfully executed, the following result is returned:

```
{
 "took" : 689,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 1,
   "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
   {
    "_index" : "my_index",
    "_type" : "_doc",
    "_id" : "2",
    "_score" : 1.0,
    "_source" : {
     "title" : "Two",
     "tags" : [
      "ruby"
     ],
     "post_date" : "2009-11-15T14:00:00"
    }
   }
  ]
 }
}
```


For more information, see open source Elasticsearch documentation.

11.2.5.5. Perform a complex search

This topic describes how to use Elasticsearch to perform a complex search.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.

2. Run the following command to perform a complex search:

```
curl -XGET http://<HOST>:<PORT>/<index_name>/_search?pretty=true -d '
{
  "query" : {
    ""
  }
}'
```

- `<HOST>` : the endpoint of the Elasticsearch cluster.
- `<PORT>` : the port of the Elasticsearch cluster. The default port is 9200.
- `<index_name>` : the name of the index.
- `query` : the request body.

> **Note** `?pretty=true` is used to make the returned result easier to understand.

Examples:

- Search based on a time range

```
curl -XGET http://localhost:9200/my_index/_search?pretty=true -d '
{
  "query" : {
    "range" : {
      "post_date" : { "from" : "2009-11-15T13:00:00", "to" : "2009-11-15T14:00:00" }
    }
  }
}'
```

If the command is successfully executed, the following result is returned:

```
{
 "took" : 1,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 2,
   "relation" : "eq"
```

```
    "relation" : "eq"
  },
  "max_score" : 1.0,
  "hits" : [
   {
    "_index" : "my_index",
    "_type" : "_doc",
    "_id" : "2",
    "_score" : 1.0,
    "_source" : {
     "title" : "Two",
     "tags" : [
      "ruby"
     ],
     "post_date" : "2009-11-15T14:00:00"
    }
   },
   {
    "_index" : "my_index",
    "_type" : "_doc",
    "_id" : "1",
    "_score" : 1.0,
    "_source" : {
     "title" : "One World",
     "tags" : [
      "ruby"
     ],
     "post_date" : "2009-11-15T13:00:00"
    }
   }
  ]
 }
}
```


- Search based on a keyword

```
curl -XGET "http://localhost:9200/my_index/_search?pretty=true" -H 'Content-Type: application
/json' -d'
{
   "query" : {
   "simple_query_string" : {
      "query": "One +(python | ruby) -Two",
      "fields": ["title^5", "tags"],
      "default_operator": "or"
   }
 }
}'
```

If the command is successfully executed, the following result is returned:

```
{
 "took" : 1,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 1,
   "relation" : "eq"
  },
  "max_score" : 2.2378674,
  "hits" : [
   {
    "_index" : "my_index",
    "_type" : "_doc",
    "_id" : "1",
    "_score" : 2.2378674,
    "_source" : {
     "title" : "One World",
     "tags" : [
      "ruby"
     ],
     "post_date" : "2009-11-15T13:00:00"
    }
   }
  ]
 }
}
```


For more information, see open source Elasticsearch documentation.

11.2.5.6. Perform statistical analytics

This topic describes how to use Elasticsearch to perform statistical analytics on documents.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.

2. Run the following command to perform statistical analytics on documents:

```
curl -XPOST http://<HOST>:<PORT>/<index_name>/_search?pretty=true -d '
{
 "size": 0,
 "aggs": {
  "group_by_tags": {
   "terms": {
    "field": "tags"
   }
  }
 }
}'
```

- `<HOST>` : the endpoint of the Elasticsearch cluster.
- `<PORT>` : the port of the Elasticsearch cluster. The default port is 9200.
- `<index_name>` : the name of the index.

> **Note** `?pretty=true` is used to make the returned result easier to understand.

Example:

```
curl -XPOST http://localhost:9200/my_index/_search?pretty=true -d '
{
 "size": 0,
 "aggs": {
  "group_by_tags": {
   "terms": {
    "field": "tags"
   }
  }
 }
}'
```

If the command is successfully executed, the following result is returned:

```
{
 "took" : 1,
 "timed_out" : false,
 "_shards" : {
  "total" : 5,
  "successful" : 5,
  "skipped" : 0,
  "failed" : 0
 },
 "hits" : {
  "total" : {
   "value" : 2,
   "relation" : "eq"
  },
  "max_score" : null,
  "hits" : [ ]
 },
 "aggregations" : {
  "group_by_tags" : {
   "doc_count_error_upper_bound" : 0,
   "sum_other_doc_count" : 0,
   "buckets" : [
    {
     "key" : "ruby",
     "doc_count" : 2
    }
   ]
  }
 }
}
```

11.2.5.7. Delete a document

This topic describes how to use Elasticsearch to delete a document.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following command to delete a document:

```
curl -XDELETE http://<HOST>:<PORT>/<index_name>/_doc/<ID>
```

- `<HOST>` : the endpoint of the Elasticsearch cluster.
- `<PORT>` : the port of the Elasticsearch cluster. The default port is 9200.
- `<index_name>` : the name of the index.
- `_doc` : the type of the index.
- `<ID>` : the ID of the document.

Example:

```
curl -XDELETE http://localhost:9200/my_index/_doc/2
```

If the command is successfully executed, the following result is returned:

```
{
  "_index" : "my_index",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 2,
  "result" : "deleted",
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "_seq_no" : 4,
  "_primary_term" : 1
}
```

For more information, see open source Elasticsearch documentation.

11.2.6. Delete an index

This topic describes how to call the delete operation for Elasticsearch to delete an index.

Procedure

1. Access your Elasticsearch cluster from the Kibana console.For more information, see Access an Elasticsearch cluster.
2. Run the following command to delete an index:

```
curl -XDELETE http://<HOST>:<PORT>/<index_name>
```

- `<HOST>` : the endpoint of the Elasticsearch cluster.
- `<PORT>` : the port of the Elasticsearch cluster. The default port is 9200.
- `<index_name>` : the name of the index.

Example:

```
curl -XDELETE http://localhost:9200/my_index
```

If the command is successfully executed, the following result is returned:

```
{
  "acknowledged" : true
}
```

For more information, see open source Elasticsearch documentation.

11.3. Manage clusters

11.3.1. Customize a cluster list

You can customize columns that are displayed in a cluster list. This makes it easy for you to view cluster information. This topic describes how to customize the columns.

Procedure

1. Log on to the Elasticsearch console.
2. In the upper-right corner of the **Elasticsearch Instances** page, click the  icon.
3. In the **Select Filters** dialog box, select columns as needed.
4. Click **OK**.
   The system then displays the cluster list based on the selected items.

11.3.2. Export a cluster list

You can export the list of Elasticsearch clusters based on selected columns. This topic describes how to export a cluster list.

Procedure

1. Log on to the Elasticsearch console.
2. In the upper-right corner of the **Elasticsearch Instances** page, click the  icon.
3. In the **Export** dialog box, specify **Export Mode** and **Translate Heading**, and select the columns that you want to export.
   - **Export Mode**: the mode used to export the cluster list. Valid values: All and Selected. Default value: All.
   - **Translate Heading**: specifies whether to translate headings. Valid values: Yes and No. Default value: Yes. If you set this parameter to Yes, exported headings are in Chinese. If you set this parameter to No, exported headings are in English, such as Instance ID and Instance Name.
   - **Custom**: Select columns that you want to export.

4. Click OK.

11.3.3. Refresh the information of a cluster

If the information of an Elasticsearch cluster, such as its status, is not refreshed in time in the console, you can manually refresh the page to obtain the up-to-date information. This topic describes how to refresh the information of an Elasticsearch cluster.

Procedure

1. Log on to the Elasticsearch console.
2. Use one of the following methods to refresh the information of a cluster:
   - On the Elasticsearch Instances page, click the icon in the upper-right corner.
   - On the Elasticsearch Instances page, find the target cluster and click its ID in the **Instance ID/Name** column. On the **Basic Information** page, click **Refresh** in the upper-right corner.

11.3.4. View the basic information of an Elasticsearch cluster

On the Basic Information page of an Elasticsearch cluster, you can view the information of the cluster, such as the ID, name, internal endpoint, and status. This topic describes how to view the basic information of a cluster.

Procedure

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. On the **Basic Information** page, view the basic information of the cluster.

Section	Parameter	Description
Basic Information section	Instance ID	The unique ID of the cluster.
	Name	The name of the cluster. By default, the name of a cluster is the same as its ID. Cluster names are configurable. You can search for clusters by name.
	Version	Valid values: 6.3.2, 6.7.0, 6.8.2, 7.2.1, and 7.4.0.
	Region	The region where the cluster resides.
	Zone	The zone where the cluster resides.
	Internal Network Address	The internal endpoint of the cluster. In a Virtual Private Cloud (VPC), you can use the internal endpoint to access the cluster.

Section	Parameter	Description
	Internal Network Port	Elasticsearch supports the following ports: ○ Port 9200 for HTTP ○ Port 9300 for TCP
Instance Statistics	Status	The status of the cluster. A cluster has the following states: Active (green), Initializing (yellow), Unhealthy (red), Paused (red), and Expired (gray).
	Created At	The time when the cluster was created.
Configuration Info	None	The configuration of the cluster. You must specify the configuration when you create the cluster. Related configuration items include Data Nodes, Dedicated Master Nodes (optional), Client Nodes (optional), and Kibana Nodes. For more information, see Elasticsearch Operations and Maintenance Guide.

11.3.5. Log on to the Kibana console

Apsara Stack Elasticsearch provides the Kibana console for you to scale your businesses. The Kibana console is a part of the Elastic ecosystem and is seamlessly integrated into Elasticsearch. This allows you to view the status of your Elasticsearch cluster in real time and manage the cluster. This topic describes how to log on to the Kibana console.

Procedure

1. Log on to the Elasticsearch console.
2. Find the target cluster and click its ID in the **Instance ID/Name** column.
3. In the left-side navigation pane, click **Data Visualization**.
4. In the **Kibana** section of the page that appears, click **Console**.
5. On the page that appears, enter the username and password and click **Log in**The username is elastic. The password is the one that is specified when the cluster is created. You can obtain the password from operations and maintenance (O&M) personnel.

11.3.6. Create a snapshot and restore data

You can call the snapshot operation to back up or restore data for your Apsara Stack Elasticsearch cluster. The snapshot operation retrieves the status and data of your cluster and stores them to a shared repository.

The first snapshot is a full copy of the data in a cluster. Subsequent snapshots store only incremental data. When you create a subsequent snapshot, the system only adds data to or removes data from the previous snapshot. This means that it requires less time to create a subsequent snapshot than the first snapshot.

Precautions

- This topic uses the following markers to provide descriptions for code: <1>, <2>, and <3>. Remove these markers before you run the code.
- You can run all the commands provided in this topic in the Kibana console of your Elasticsearch cluster. For more information, see Log on to the Kibana console.

Create a repository

```
PUT _snapshot/my_backup
{
  "type": "oss",
   "settings": {
      "endpoint": "http://oss-cn-hangzhou-internal.aliyuncs.com", <1>
      "access_key_id": "xxxx",
      "secret_access_key": "xxxxxx",
      "bucket": "xxxxxx", <2>
      "compress": true,
      "base_path": "snapshot/" <3>
   }
}
```

- <1>: The `endpoint` parameter specifies the internal endpoint of an Object Storage Service (OSS) bucket.
- <2>: The bucket parameter specifies the name of the OSS bucket.
- <3>: The base_path parameter specifies the path of the repository. The default value is the root directory.

Set the size of each part

You can upload large volumes of data to an OSS bucket in multiple parts. When you upload the data, you can use the `chunk_size` parameter to set the size of each part. Example:

```
POST _snapshot/my_backup/ <1>
{
   "type": "oss",
   "settings": {
      "endpoint": "http://oss-cn-hangzhou-internal.aliyuncs.com",
      "access_key_id": "xxxx",
      "secret_access_key": "xxxxxx",
      "bucket": "xxxxxx",
      "chunk_size": "500mb",
      "base_path": "snapshot/" <2>
   }
}
```

- <1>: Use the POST method instead of the PUT method. The POST method updates repository settings.
- <2>: The base_path parameter specifies the path of the repository. The default value is the root directory.

Query repository information

```
GET _snapshot
```

You can also use the `GET _snapshot/my_backup` command to query the information of a specified repository.

Create a snapshot for all open indexes

The following command is a basic command that is used to create a snapshot:

```
PUT _snapshot/my_backup/snapshot_1
```

This command creates a snapshot named `snapshot_1` for all open indexes and stores the snapshot in a repository named `my_backup`. After you run the command, the system immediately returns a response and creates the snapshot at the backend.

If you want the system to return a response after the snapshot is created, specify the `wait_for_completion` parameter in the command. Example:

```
PUT _snapshot/my_backup/snapshot_1?wait_for_completion=true
```

After you run the command, the system does not return a response until the snapshot is created. If the size of an index is large, the response is returned after a longer period of time.

Create a snapshot for specified indexes

By default, a snapshot contains all open indexes. If Kibana is used when you create a snapshot, you may want to ignore all diagnostic indexes (the `.kibana` indexes) because of limited disk space. To create a snapshot for specified indexes, run the following command.

> **Notice** A repository stores multiple snapshots. Each snapshot is a copy of all indexes, specified indexes, or a single index in a cluster. When you create a snapshot, make sure that the snapshot name is unique.

```
PUT _snapshot/my_backup/snapshot_2
{
  "indices": "index_1,index_2"
}
```

The preceding command creates a snapshot only for the `index1` and `index2` indexes.

Query snapshot information

In some cases, you may need to query snapshot information. For example, a snapshot name that contains a date is hard to remember, such as `backup_2014_10_28`.

To query the information of a snapshot, send a `GET` request that contains both the repository name and snapshot name. Example:

```
GET _snapshot/my_backup/snapshot_2
```

The response contains detailed information of the snapshot:

```
{
"snapshots": [
  {
    "snapshot": "snapshot_2",
    "indices": [
      ".marvel_2014_28_10",
      "index1",
      "index2"
    ],
    "state": "SUCCESS",
    "start_time": "2014-09-02T13:01:43.115Z",
    "start_time_in_millis": 1409662903115,
    "end_time": "2014-09-02T13:01:43.439Z",
    "end_time_in_millis": 1409662903439,
    "duration_in_millis": 324,
    "failures": [],
    "shards": {
      "total": 10,
      "failed": 0,
      "successful": 10
    }
  }
]
}
```

You can replace the snapshot name in the preceding command with `_all` to query all snapshots in the repository. Example:

```
GET _snapshot/my_backup/_all
```

Delete a snapshot

You can specify a repository name and a snapshot name in a `DELETE` request to delete the specified snapshot. Example:

```
DELETE _snapshot/my_backup/snapshot_2
```

> **Notice**
>
> - You can delete snapshots only by calling the DELETE operation. A snapshot is associated with other backup files. Some of the files may also be used by other snapshots. The DELETE operation does not delete files that are still being used by other snapshots. It deletes only the files that are associated with deleted snapshots and are no longer used by other snapshots.
> - If you choose to manually delete a snapshot, you may delete files that are used by other snapshots. This can cause data loss.

Monitor snapshot creation progress

The `wait_for_completion` parameter provides a simple method for you to monitor the progress of a snapshot creation task. However, this parameter is not suitable for snapshot creation tasks of medium-size Elasticsearch clusters. You can use one of the following methods to query detailed information about a snapshot:

- Send a `GET` request with the snapshot name specified. Example:

  ```
  GET _snapshot/my_backup/snapshot_3
  ```

  If the system is creating the snapshot when you run the preceding command, the information of the creation task is returned, such as the start time and duration of the task.

  > **Notice** The preceding command shares a thread pool with the command used to create a snapshot. Therefore, if you create a snapshot for large shards, the preceding command has to wait until the resources that are used by the snapshot creation command in the thread pool are released.

- Call the status operation to query the snapshot status.

  ```
  {
  "snapshots": [
    {
      "snapshot": "snapshot_3",
      "repository": "my_backup",
      "state": "IN_PROGRESS", <1>
      "shards_stats": {
        "initializing": 0,
        "started": 1, <2>
        "finalizing": 0,
        "done": 4,
  ```

```
    "failed": 0,
    "total": 5
  },
  "stats": {
    "number_of_files": 5,
    "processed_files": 5,
    "total_size_in_bytes": 1792,
    "processed_size_in_bytes": 1792,
    "start_time_in_millis": 1409663054859,
    "time_in_millis": 64
  },
  "indices": {
    "index_3": {
      "shards_stats": {
        "initializing": 0,
        "started": 0,
        "finalizing": 0,
        "done": 5,
        "failed": 0,
        "total": 5
      },
      "stats": {
        "number_of_files": 5,
        "processed_files": 5,
        "total_size_in_bytes": 1792,
        "processed_size_in_bytes": 1792,
        "start_time_in_millis": 1409663054859,
        "time_in_millis": 64
      },
      "shards": {
        "0": {
          "stage": "DONE",
          "stats": {
            "number_of_files": 1,
            "processed_files": 1,
            "total_size_in_bytes": 514,
            "processed_size_in_bytes": 514,
            "start_time_in_millis": 1409663054862,
            "time_in_millis": 22
          }
        },
```

```
            ...
```

- <1>: the status of the snapshot. If a snapshot is being created, the value of the field is `IN_PROGRESS`.
- <2>: the number of shards that are being transmitted. If the value 1 is returned, a shard is being transmitted to the snapshot, and the other four shards have been transmitted.

  The value of the `shards_stats` parameter contains the status of the snapshot and the statistics about each index and shard. This parameter allows you to obtain the detailed information of the snapshot creation progress. A shard can be in one of the following states:

  - `INITIALIZING`: The shard is verifying the status of the cluster to check whether the shard can be stored in a snapshot. In most cases, this process is fast.
  - `STARTED`: Data is being transmitted to the repository.
  - `FINALIZING`: Data is transmitted, and the shard is sending snapshot metadata.
  - `DONE`: A snapshot is created for the shard.
  - `FAILED`: An error occurred during the snapshot creation. The shard, index, or snapshot cannot be processed. You can view logs for more information.

Cancel a snapshot

To cancel a snapshot, run the following command when the snapshot is being created:

```
DELETE _snapshot/my_backup/snapshot_3
```

This command stops the snapshot creation process and deletes the snapshot that is being created from the repository.

Restore indexes from a snapshot

To restore indexes from a snapshot, run the command that is used in the "" section on the Elasticsearch cluster that stores these indexes. You can use one of the following methods to restore indexes from a snapshot:

- To restore indexes from a specified snapshot, append the `_restore` parameter to the snapshot name in the command to run. Example:

  ```
  POST _snapshot/my_backup/snapshot_1/_restore
  ```

  After you run this command, the system restores all indexes in the snapshot. For example, if the `snapshot_1` snapshot contains five indexes, all these indexes are restored to the related Elasticsearch cluster. You can also specify the indexes that you want to restore. For more information, see the "" section.
- Restore specified indexes and rename the indexes. If you only want to verify or process the data in indexes and do not want to overwrite the data, use this method to restore the indexes.

```
POST /_snapshot/my_backup/snapshot_1/_restore
{
 "indices": "index_1", <1>
 "rename_pattern": "index_(.+)", <2>
 "rename_replacement": "restored_index_$1" <3>
}
```

In this example, the `index_1` index is restored to your Elasticsearch cluster and renamed `restored_index_1`.

- <1>: The system restores only the `index_1` index from the snapshot.
- <2>: The system searches for the index that is being restored and matches the index name with the provided index pattern.
- <3>: The system renames the matched index.

- If you want the system to return a response after it restores the index, specify the `wait_for_completion` parameter in the command. Example:

  ```
  POST _snapshot/my_backup/snapshot_1/_restore?wait_for_completion=true
  ```

  After you call the restore operation, the system immediately returns a response and restores the index at the backend. If you want the system to return a response after the index is restored, specify the `wait_for_completion` parameter.

Monitor index restoration progress

> **Note** Restoring data from a repository applies the existing restoration mechanism in Elasticsearch. Restoring shards from a repository is the same as restoring data from a node.

You can call the recovery operation to monitor the progress of an index restoration task.

- Monitor the restoration of a specified index.

  ```
  GET restored_index_3/_recovery
  ```

  The recovery operation is a general-purpose operation that shows the status of the shards that are being transmitted to your cluster.

- Monitor the restoration of all indexes on the cluster. This may include shards that are irrelevant to the restoration process.

  ```
  GET /_recovery/
  ```

  Sample response:

  ```
  {
  "restored_index_3" : {
   "shards" : [ {
    "id" : 0,
  ```

```
  "type" : "snapshot", <1>
  "stage" : "index",
  "primary" : true,
  "start_time" : "2014-02-24T12:15:59.716",
  "stop_time" : 0,
  "total_time_in_millis" : 175576,
  "source" : { <2>
    "repository" : "my_backup",
    "snapshot" : "snapshot_3",
    "index" : "restored_index_3"
  },
  "target" : {
    "id" : "ryqJ5lO5S4-lSFbGntkEkg",
    "hostname" : "my.fqdn",
    "ip" : "10.0. **.**",
    "name" : "my_es_node"
  },
  "index" : {
    "files" : {
      "total" : 73,
      "reused" : 0,
      "recovered" : 69,
      "percent" : "94.5%" <3>
    },
    "bytes" : {
      "total" : 79063092,
      "reused" : 0,
      "recovered" : 68891939,
      "percent" : "87.1%"
    },
    "total_time_in_millis" : 0
  },
  "translog" : {
    "recovered" : 0,
    "total_time_in_millis" : 0
  },
  "start" : {
    "check_index_time" : 0,
    "total_time_in_millis" : 0
  }
}]
```

```
}
}
```

- <1>: The `type` parameter indicates the type of restoration. The value `snapshot` indicates that the shard is being restored from a snapshot.
- <2>: The `source` parameter indicates the source snapshot and repository.
- <3>: The `percent` parameter indicates the progress of the restoration task. The value `94.5%` indicates that 94.5% of the shard files are restored.

The response lists all indexes that are being restored and the shards in these indexes. Each shard has statistics about the start or end time, duration, restoration progress, and bytes transmitted.

Cancel index restoration

To cancel index restoration, you only need to delete the indexes that are being restored. A restoration process is a shard restoration process. You can call the DELETE operation to modify the status of a cluster and cancel index restoration. Example:

```
DELETE /restored_index_3
```

If you run the preceding command when the `restored_index_3` index is being restored, the system stops the restoration and deletes the data that has been restored to your cluster. For more information, see Snapshot And Restore.

Use a snapshot to migrate data

To use a snapshot to migrate data from an Elasticsearch cluster to another, perform the following steps:

1. Back up a snapshot to OSS.
2. Create a snapshot repository on the destination cluster. The repository must use the OSS bucket that stores the snapshot.
3. Set the `base_path` parameter to the path of the snapshot.
4. Run the data restoration command on the destination cluster.

> **Note** These steps provide a simple solution to snapshot-based data migration. For more information, see Use a snapshot stored in OSS to migrate Elasticsearch data.

11.4. Use plug-ins

11.4.1. Use the a-pack-xdcr plug-in to replicate data across Elasticsearch clusters

This topic describes how to use the a-pack-xdcr plug-in for Elasticsearch to replicate data across Elasticsearch clusters in different data centers. This implements remote disaster recovery. Only Elasticsearch V6.7.0 and V6.8.2 clusters support this plug-in.

Prerequisites

- The a-pack-xdcr plug-in is installed on the source and destination Elasticsearch clusters.

  You can contact O&M personnel to install the plug-in. For more information, see Elasticsearch Operations and Maintenance Guide.

- The IP address of the source Elasticsearch cluster is obtained.

  You can obtain the IP address and port number of the source Elasticsearch cluster by running the `kubectl get svc -n <elasticsearch namespace>` command on the relevant Kubernetes server. You can also obtain the information from O&M personnel.

Context

Elasticsearch provides the cross data center replication (XDCR) feature. It allows you to replicate data across Elasticsearch clusters to implement remote high availability (HA). Core link applications must be able to withstand the impact of service interruptions in data centers or regions. The native capabilities of XDCR can meet disaster recovery and HA requirements across data centers without requiring additional technologies. You can use the XDCR feature to replicate data to data centers that are located in close proximity to users or application servers. This reduces latency and costs. For example, you can replicate product lists or reference datasets to 20 or more data centers around the world to minimize the distance between data and application servers.

Procedure

1. Log on to the Kibana console of the destination Elasticsearch cluster.For more information, see Log on to the Kibana console.
2. In the left-side navigation pane, click **Dev Tools**.
3. On the **Console** tab, run the following command to create a remote repository in the destination Elasticsearch cluster and connect the repository to the source Elasticsearch cluster:

```
PUT _cluster/settings
{
"persistent": {
"cluster": {
    "remote": {
      "leader": {
        "seeds": [
          "<your cluster IP>:<your cluster port>"
        ]
      }
    }
}
}
}
```

Note Replace <your cluster IP> with the IP address of the source cluster. Replace <your cluster port> with the port number of the source cluster. In most cases, the port number is 9300.

4. Run the following command to replicate data in an index:

```
PUT _xdcr/leader/<your_index>
```

Note Replace <your_index> with the name of the index whose data you want to replicate.

5. Run the following command to view the status of the replication task:

```
GET _cat/xdcr?v
```

If the command is successfully executed, the following result is returned.

index	repository	shard	localSeqNo	remoteSeqNo
twitter	leader	4	1	1
twitter	leader	1	0	0
twitter	leader	2	7	7
twitter	leader	3	3	3
twitter	leader	0	-1	-1

> **Note** index indicates the index name. repository indicates the destination cluster. shard indicates the index shard. localSeqNo indicates the maximum sequence number of the source cluster. remoteSeqNo indicates the maximum sequence number of the destination cluster.

6. Run the following command to view replicated data:

```
GET /<your_index>/_search
```

> **Note** Replace `<your_index>` with the name of the index that stores the replicated data.

11.4.2. Use the opendistro_sql plug-in to query cluster data by executing SQL statements

This topic describes how to use the opendistro_sql plug-in for Elasticsearch to query data in an Elasticsearch cluster by executing SQL statements.

Prerequisites

The opendistro_sql plug-in is installed. You can contact O&M personnel to install the plug-in. For more information, see Elasticsearch Operations and Maintenance Guide.

Procedure

1. Log on to the Kibana console of your Elasticsearch cluster.For more information, see Log on to the Kibana console.
2. In the left-side navigation pane, click **Dev Tools**.
3. On the **Console** tab, run one of the following commands to query cluster data by executing SQL statements.Elasticsearch supports the following SQL query syntax:

> **Note** Replace `<your_index>` in the following code with the name of the index whose data you want to query.

   - Syntax for an SQL query in a GET request

```
GET _opendistro/_sql?sql=select * from <your_index> limit 50
```

   - Syntax for an SQL query in a POST request

```
POST _opendistro/_sql
{
 "query": "SELECT * FROM <your_index> LIMIT 50"
}
```

   - Syntax for an SQL query with the CSV format specified for the data to return

```
POST _opendistro/_sql?format=csv
{
 "query": "SELECT * FROM <your_index> LIMIT 50"
}
```

Note

- For more information about SQL syntax, see SQL.
- If you want to execute Java code by using SQL statements, download the JDBC driver at https://github.com/opendistro-for-elasticsearch/sql-jdbc.

11.4.3. Use the bsearch_querybuilder plug-in to construct query conditions

This topic describes how to use the bsearch_querybuilder plug-in for Elasticsearch to construct query conditions in Kibana. This plug-in implements visualized data queries. Only Kibana V6.7.0 and V6.8.2 support this plug-in.

Prerequisites

The bsearch_querybuilder plug-in is installed. You can contact O&M personnel to install the plug-in. For more information, see Elasticsearch Operations and Maintenance Guide.

Procedure

1. Log on to the Kibana console of your Elasticsearch cluster.For more information, see Log on to the Kibana console.
2. In the left-side navigation pane, click **Discover**. On the page that appears, click **Query** in the upper-right corner.

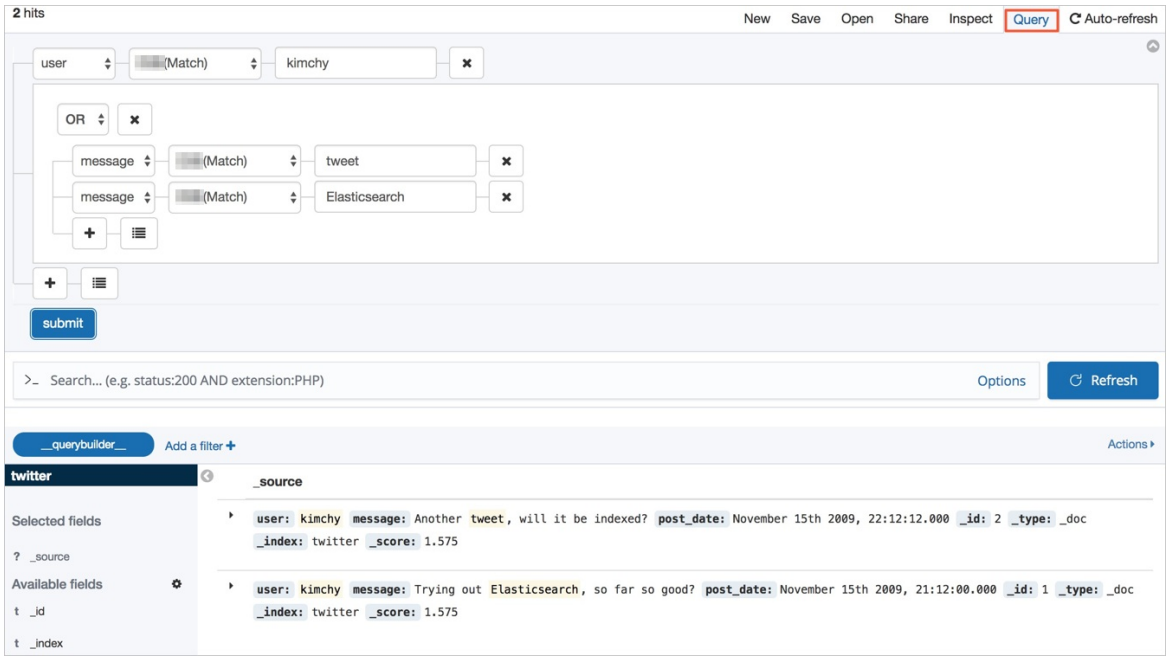
   Notice

   - Before you create a query, make sure that you have created an index pattern. To create an index pattern in the Kibana console, click **Management** in the left-side navigation pane. On the page that appears, click **Index Patterns** in the Kibana section. Then, click **Create index pattern** and create an index pattern as prompted.
   - If the bsearch_querybuilder plug-in is installed, but **Query** is not displayed in the Kibana console, the plug-in may not take effect. In this case, you must run the `kubectl delete deployments elasticsearch-cluster-kibana -n <your_es_namespace>` command on the relevant Kubernetes server to restart Kibana.

3. On the page that appears, specify query conditions and filters.Click the + icon to add a query condition. Click the ☰ icon to add a filter for the query condition. Click the ✖

icon to delete a query condition or filter.

The following figure shows a query example. In this example, the user condition is set to kimchy, and the type filters are set to tweet and Elasticsearch.



4. Click **submit**.

11.4.4. Use alerting plug-ins to implement alerting for an Elasticsearch cluster

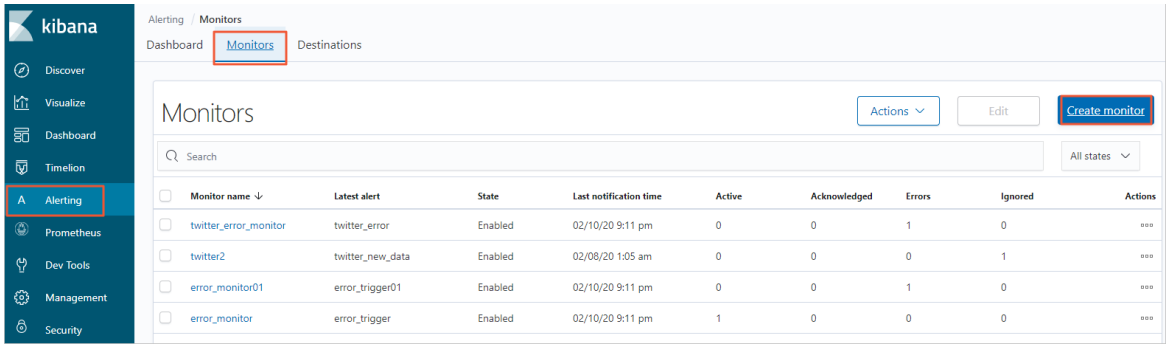
This topic describes how to use the alerting plug-ins of Elasticsearch and Kibana to implement alerting for an Elasticsearch cluster. You can configure alerts and customize alert thresholds and conditions for the cluster. The system sends alert notifications if the alert conditions are met.

Prerequisites

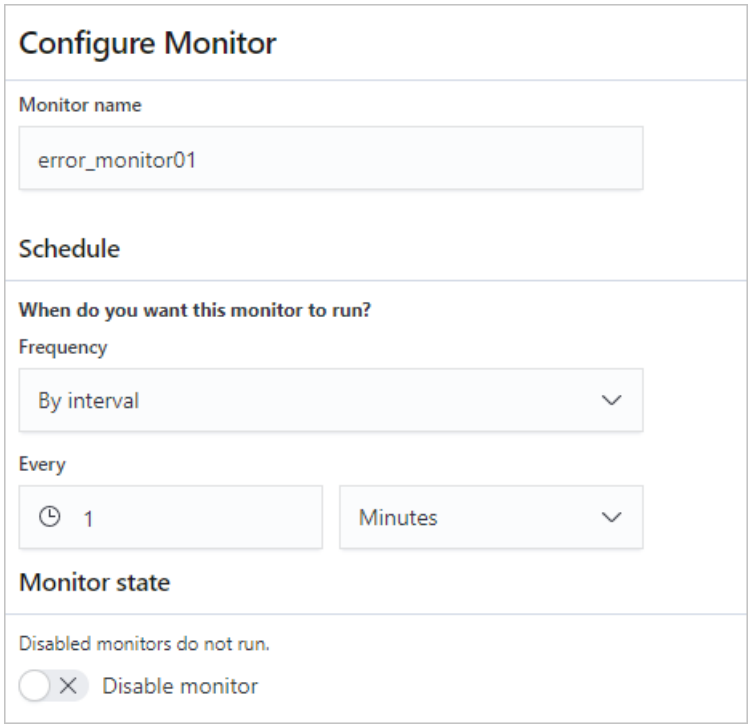
The opendistro_alerting plug-in for Elasticsearch and the opendistro-alerting plug-in for Kibana are installed. You can contact O&M personnel to install the plug-ins. For more information, see Elasticsearch Operations and Maintenance Guide.

Procedure

1. Log on to the Kibana console of your Elasticsearch cluster.For more information, see Log on to the Kibana console.
2. In the left-side navigation pane, click **Alerting**.
3. On the page that appears, click the **Monitors** tab.
4. In the upper-right corner of the tab, click **Create Monitor**.



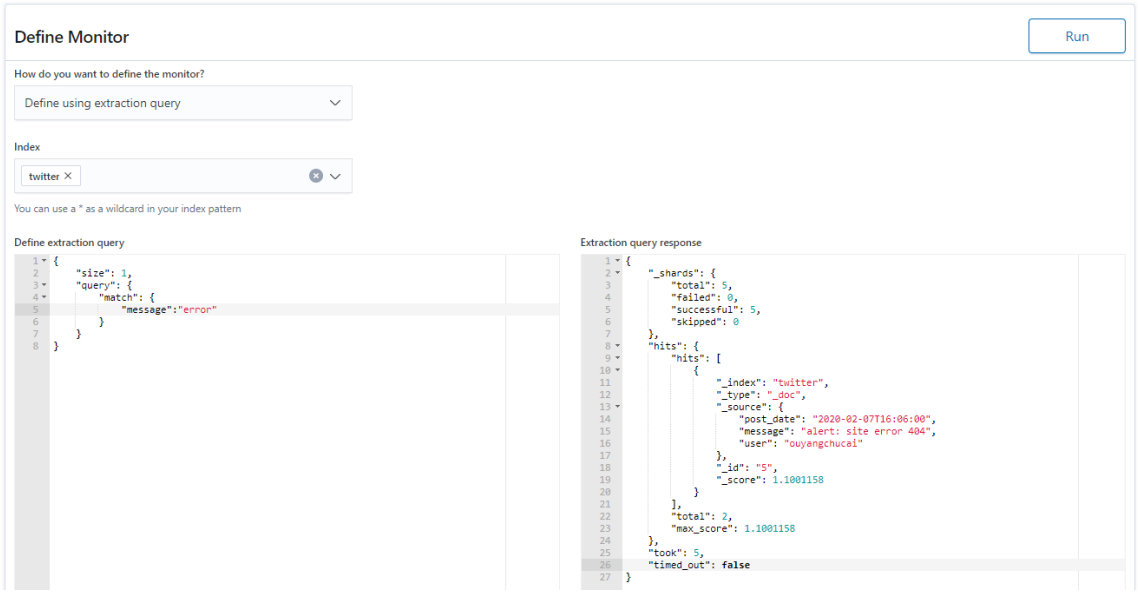
5. On the page that appears, specify the required parameters.



In this example, the name of the monitor is error_monitor01, and the monitor runs once a minute.

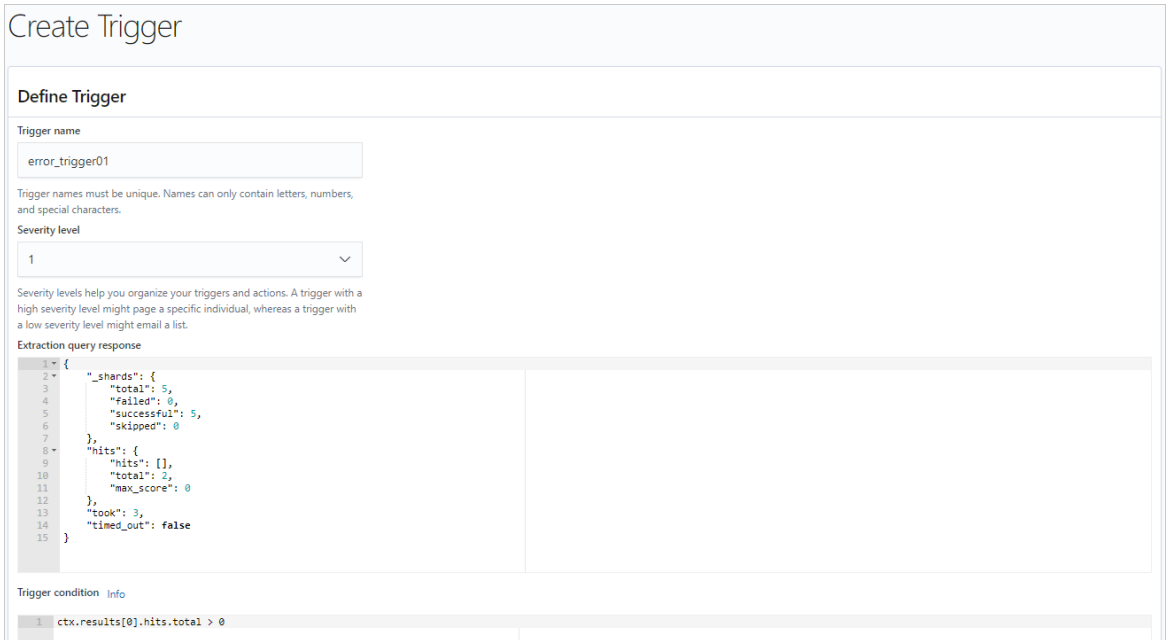
6. Define the monitor.Elasticsearch allows you to use visual images or specific query statements to define a monitor.
   - If you want to use visual images to define the monitor, you must specify an **index** and a **time field**.
   - If you want to use specific query statements to define the monitor, specify an **index** and query conditions. Then, click **Run** and view results. This method is used in this topic.

The following figure shows the query condition. This query is to obtain data that includes the error keyword from messages.



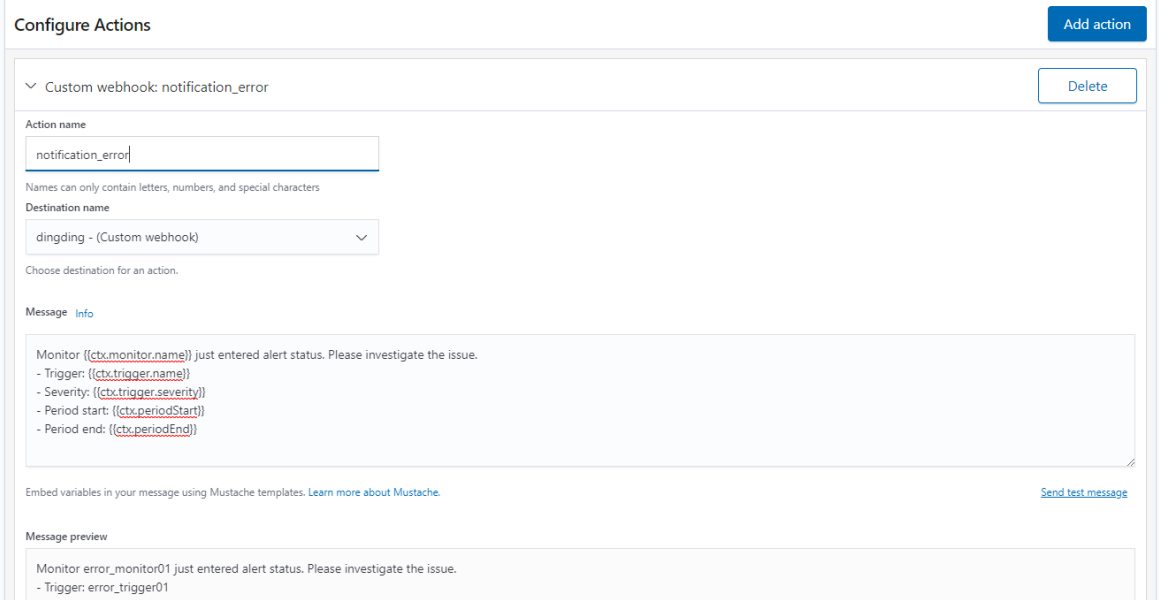

```
{
  "size": 1,
  "query": {
    "match": {
      "message":"error"
    }
  }
}
```

7. Click **Create**.
8. On the **Create Trigger** page, define and configure an alert trigger.



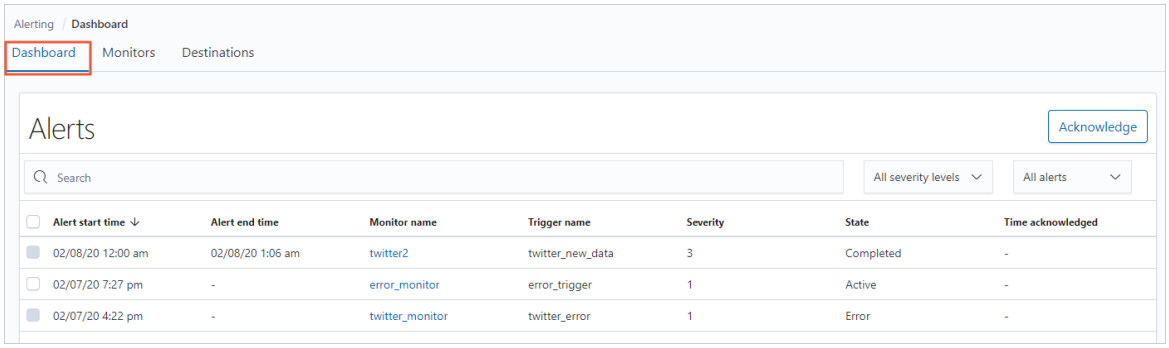
Parameter	Description
Trigger name	The name of the trigger.
Severity level	The severity of the trigger.
Extraction query response	The query response. This is specified in the alert definition and cannot be modified.
Trigger condition	The condition to trigger the alert. You can modify the condition based on your business requirements. For example, the condition is `ctx.results[0].hits.total > 0`. This indicates that an alert is reported when the number of returned entries is greater than 0.

9. Configure an alert notification.In the **Configure Actions** section, specify the required parameters.



Parameter	Description
Action name	The name of the notification.
Destination name	The destination address of the notification. In most cases, the address is a URL.
Message	The content of the notification. The content can be modified based on your requirements.
Message preview	The notification preview. This cannot be modified.

10. Click **Create**.
11. On the Alerting page, click the **Dashboard** tab to view the newly created alert.


Alerting / Dashboard
Dashboard
Monitors
Destinations
Alerts
Acknowledge
Search
All severity levels
All alerts
Alert start time
Alert end time
Monitor name
Trigger name
Severity
State
Time acknowledged
02/08/20 12:00 am
02/08/20 1:06 am
twitter2
twitter_new_data
3
Completed
-
02/07/20 7:27 pm
-
error_monitor
error_trigger
1
Active
-
02/07/20 4:22 pm
-
twitter_monitor
twitter_error
1
Error
-